HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

BACHELORARBEIT

Herr
Michael Jope

Entwicklung eines Web und Android gestützten CRM Systems

2011

Fakultät **Mathematik/Naturwissenschaften/Informatik**

BACHELORARBEIT

Entwicklung eines Web und Android gestützten CRM Systems

Autor:
Michael Jope

Studiengang:
Informatik

Seminargruppe:
IF08w1-B

Erstprüfer:
Prof. Dr. Mario Geißler

Zweitprüfer:
Dipl. Ing.(FH) Felix Trägner

Mittweida, 2011

Bibliografische Angaben

Jope, Michael: Entwicklung eines Web und Android gestützten CRM Systems, 47 Seiten, 17 Abbildungen, Hochschule Mittweida (FH), Fakultät Mathematik/Naturwissenschaften/Informatik

Bachelorarbeit, 2011

Referat

Ziel dieser Bachelorarbeit ist die Entwicklung eines CRM Systems. Da für einzelne Abläufe die Mobilität und Flexibilität im Vordergrund steht, sind sie auf der Android Plattform umzusetzen. Das Hauptsystem jedoch ist als Web Anwendung konzipiert.

Die Umsetzung der vorgegebenen Abläufe, sowie die vorherige Planung bildet einen großen Teil der Arbeit.

Da das Programm private Kundendaten verarbeitet sind einige Sicherheitsaspekte zu berücksichtigen. Diese werden in der Arbeit vom Serverbetriebssystem bis hin zur skriptgesteuerten Erstellung der Webinhalte betrachtet.

Es werden einige Teile der Implementierung vorgestellt. Diese Codeabschnitte zeichnen sich durch im hohen Maße für den Systemablauf wichtige Implementierungen von Funktionen aus.

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die mich bei der Anfertigung dieser Arbeit unterstützten.
Allen voran bei meinem Betreuern Herrn Professor Dr. Geißler und Herrn Dipl. Ing.(FH) Felix Trägner, die mich fachkundig unterstützten.

Ein weiterer Dank gilt Herrn Andreas Lux, der mir viele Abläufe erläuterte und neue Ideen zu den Modulen Beitrug. In gleichen Atemzug möchte ich mich bei Frau Anna Povazhuk für ihre russisch Übersetzung bedanken.

Ein großes Dankeschön gilt auch allen Korrekturlesern.

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Tabellenverzeichnis

IV. Abkürzungsverzeichnis

ACL Access Control List, dt. Zugriffskontrolllisten 23

AJAX Asynchronous JavaScript and XML 35

apt advanced packaging toolkit 8

CRM Customer Relationship Mangement, dt. Kundenbeziehungsmanagement

CSS Cascading Style Sheets 9

DOS Denial of Service, dt. Ablehnung des Service 23

FLASK Flux Advanced Security Kernel 21

FTP File Transfer Protocol 24

HTML Hypertext Markup Language 9

HTTP Hyper Text Transfer Protokoll 6

HTTPS Hyper Text Transfer Protokoll Secured 6

JDK Java Development Kit 4

JSON Java Script Object Notation 35

LAMP Linux Apache MySQL PHP/Perl/Python 8

PHP Personal Homepage Tools V

SDK Software Development Kit 4

SELinux Security Enhanced Linux, dt. erweiterte Sicherheit für Linux 10

SFTP SSH File Transfer Protocol 24

SQL Structured Query Language 25

SSH secure shell 8

SSL Secure Sockets Layer

TLS Transport Layer Security

URL Uniform Resource Locator, dt. Einheitlicher Quellenanzeiger 6

USB Universal Serail Bus 9

XML Extensible Markup Language 7

XSS Cross Site Scripting 25

yum yellowdog updater modified 8

V. Quellcodeverzeichnis

VI. Vorwort

Diese Bachelorarbeit entstand in dem Unternehmen **EASTCON Produktionsgesellschaft**. Der Sitz des Unternehmens ist in Rochlitz, wo es im Jahr 2003 als Tochterunternehmen der **EASTCON AG** gegründet wurde. Es fertigt Güter wie beispielsweise Ober- und Unterbetten aus Naturhaarmaterialien. Die fertigen Produkte gehen an die 100%igen Tochtergesellschaften der Eastcon AG in den verschiedenen osteuropäischen Ländern. Diese setzen sie dann in ihren lokalen Märkten ab. Im Vertriebsprozess wird der Kunde angesprochen, die Ware präsentiert und die Verkäufe vertraglich festgehalten. Kundendaten werden dabei in Listen festgehalten, die allerdings nur für eine einmalige Geschäftstransaktion verwendet werden. Für qualifizierende Auswertungen ist ein erheblicher manueller Mehraufwand nötig. Daten sind nicht systemisch erfasst und auswertbar.

Diese Arbeit befasst sich mit dem Erstellen und Sichern des Prototyps eines CRM Systems. Hierbei werden erst einmal die Schritte von der Datengewinnung bis hin zur Teilname des Interessenten an der ersten Verkaufsveranstaltung betrachtet. Als Grundlage wurden Beschreibungen bestehender Abläufe verwendet. Diese lassen sich in drei Phasen einteilen:

1. Sammeln von Kundenadressen

2. Planen von Veranstaltungen

3. Interessenten einladen

Diese werden in dem System abgebildet. Weiter wird hier behandelt welche Maßnahmen ergriffen werden sollten, um das System zu schützen. Das System hat die Auflage leicht erweiterbar zu sein, um neue Module erstellen können. Es ist dabei verantwortlich für das Pflegen einer Datenbank als Backend.

1 Einleitung

1.1 Aufgabenstellung

Diese Arbeit entstand in einem Unternehmen, welches große Teile seines Gewinnes im Direktverkauf erziehlt. Dies erfordert eine kundenbezogene Arbeitsweise. Um solche Prozesse zu unterstützen, soll eine Software entwickelt werden, die idealerweise folgende Funktionalitäten besitzt:

- Stationär ungebundene Adresssammlung: Dem Adresssammler wird die Möglichkeit gegeben die Interessentendaten von stationär gebundenen Systemen unabhängig zu sammeln. Jedoch sind dabei Sammlungen in Papierform nicht zulässig, so wie zur Zeit noch vorrangig durchgeführt.

- Multi User Eigenschaften: Eine Prämisse, die bei der Entwicklung zu berücksichtigen ist, ist die Verfügbarkeit des Systems für viele gleichzeitig angemeldete Mitarbeiter rund um die Uhr.

- Webfähigkeit: Mitarbeiter wird damit ermöglicht Home Office zu betreiben.

- Sicherheit: Das System muss bestmöglich gegen Eindringlinge und unsach-gemäße Handhabung geschützt werden.

- Bedienbarkeit: Es sind bestehenden Abläufe abzubilden. Die Orientierung an diesen Vorgehensweisen, gibt den Mitarbeitern einen schnellen Einstieg in den Umgang mit dem System.

- Multilingualität: Da das System in verschiedenen Ländern zum Einsatz kommt, muss es eine umfassende Sprachunterstützung bekommen. Dabei sind auch die verschiedenen Eigenheiten der Länder berücksichtigt werden, wie zum Beispiel der Namenszusatz Vatersname in vielen osteuropäischen Länder.

In dieser Arbeit wird erklärt, wie die Umsetzung einer solchen Software mit den betrieblichen Vorgaben zu bewerkstelligen ist. Die Erstellung soll dabei Iterativ erfolgen um Verbesserungsvorschläge und Änderungen im Ablauf schnell zu berücksichtigen.

1.2 Motivation

Die Neuerungen im Bereich der mobilen Endgeräte, sowie die webbasierte Umsetzung des Systems waren die Haupmotivationspunkte, die zu dieser Arbeit veranlassten. Auch

das Einbringen von Technologien, zum einen zur Verbesserung der Bedienbarkeit, zum anderen zum Bereitstellen von essenziellen Funktionen des Gesamtsystems, war ein weiterer Punkt. Ein interessanter Aspekt war auch die Implementierung dieser Techniken. Die betriebswirtschaftlichen Hintergründe zu erfassen und einzubringen, war ein weiterer Problematik die das Interesse zur Bearbeitung dieses Themas weckte.

2 Grundlagen

Dieses Kapitel befasst sich mit den Grundlagen zu dem Begriff des CRM Systems, den eingesetzten Bedienungsplattformen und den Übertragungsprotokollen. Zum Abschluss wird noch eine Anleitung zur Installation und Konfiguration von Entwicklungs und Produktivsystemen vorgestellt.

2.1 CRM System

Ein CRM System ist darauf ausgelegt, die Beziehungen zwischen Kunden und der Firma abzubilden. Dabei soll es die Mitarbeiter bei dem Umgang mit den Kunden und den mit ihm in Beziehung stehenden Abläufen unterstützen.

Abgrenzung von anderen Systemen: Das Kundenbeziehungsmanagement entwickelte sich direkt aus dem Beziehungsmarketing, nicht zu verwechseln mit dem, was noch horizontale (z.B. Vertriebsgemeinschaften), vertikale (z.B. Lieferanten), laterale (z.B. Beziehungen zu Behörden) und unternehmerische Beziehungen (z.B. zum Personal) mit einbezieht. Kundenbindungsmanagement kann wiederum als ein Teil des Beziehungsmarketing verstanden werden, es beschäftigt sich nur mit dem gegenwärtigen Kundenstamm, mit dem Ziel, dass sie die Beziehungen zum Unternehmen beibehalten, festigen oder gar weiter ausbauen. Was zum Kundenbeziehungsmanagement noch fehlt, sind Konzepte für die Neukunden- und Kundenrückgewinnung. CRM wurde im Marketing schon lange angewendet, durch Internet und informationstechnische Unterstützung wurde es mit elektronischen Mitteln umgesetzt. Dabei erfolgt die Umsetzung oft als Web-Anwendung.[vgl. S. 18-19 7, Kap 2.2]

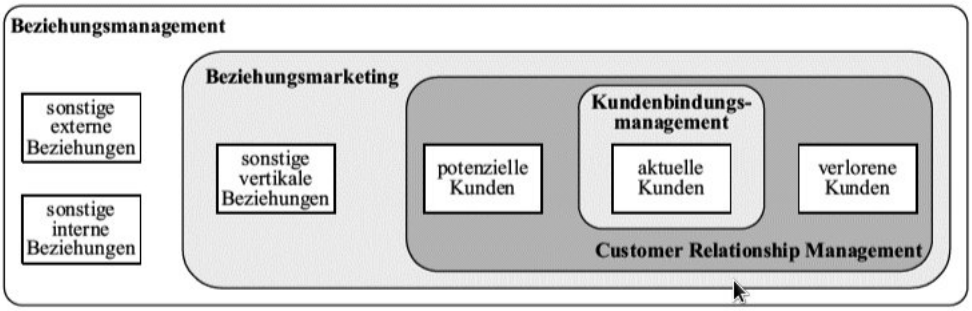

Abbildung 2.1: Übersicht CRM *Quelle: [7, S. 20]*

Durch das rasante Wachstum des Internet und der Etablierung der elektronischen Datenverarbeitung, wurden diese Strukturen für Mehrnutzerprogramme immer attraktiver. In diesem Zusammenhang wurden auch die Methoden des Kunden-

beziehungsmanagement auf Software abgebildet. Überwiegend geschah diese Umsetzung in Webanwendungen (vgl. Abschnitt 2.3). Diese Programme werden CRM-Systeme genannt.

2.2 Android Plattform

Android ist ein freies Betriebssystem für Mobile Geräte, als Solches ist es sehr darauf bedacht, Ressourcen zu sparen.

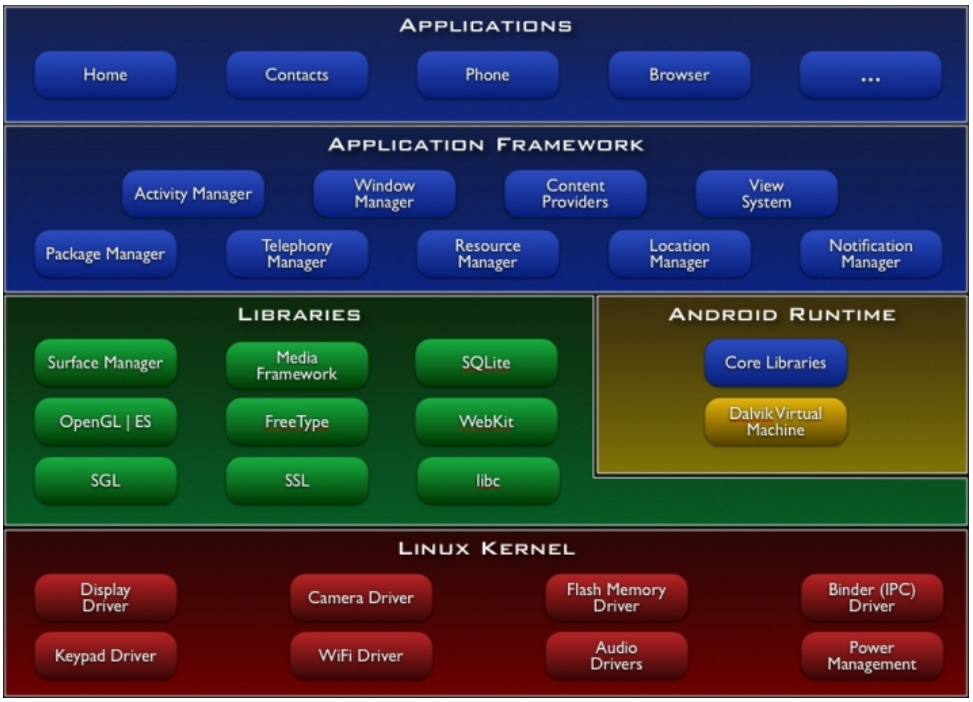

Abbildung 2.2: Android Systemarchitektur *Quelle: [6]*

Durch den in Abbildung 2.2 dargestellten Aufbau, ist das System komplett auf mobile Endgeräte abgestimmt und auch gegen Angriffe geschützt. Die Bestandteile und deren Funktion werden in Tabelle 2.1 umschrieben. Die Java Laufzeitumgebung wird auch durch Prozessoren unterstützt, welche den Bytecode nativ ausführen können.[vgl. 1]

Die Dokumentation des Software Development Kit (SDK) ist vergleichbar mit der Dokumentation des Java Development Kit (JDK). Mithilfe dieses SDK ist es möglich, Software für das Android System zu entwickeln. Die entstandenen Java Applikationen werden hier nicht zwangsläufig über den **Android Market** verteilt (vgl. Abschnitt 2.5). Dem Entwickler bietet das System dabei folgende Funktionsmerkmale [vgl. 6]:

- Anwendungs Framework: Mit ersetzbaren und veränderbaren Bestandteilen
- Integrierter Browser: Auf der Webkit Engine basierenter Browser

- Optimierte Grafik: Eine selbst entwickelte 2D Bibliothek; OpenGL ES Spezifikation für die 3D Bibliothek
- SQLite: Zum strukturierten Speichern von Daten
- Multimedia: Unterstützung für die gebräuchlichsten Formate(MP3, JPG, usw.)
- Hardwareabhängige Funktionen (z.B. GPS, WLAN, usw.)

Tabelle 2.1: Systemkomponenten die Android laut GOOGLE SDK [6] zur Verfügung stellt.

Applications	Sind Java-Applikationen, ohne die Interaktion mit dem Gerät nicht denkbar wäre. Ein Beispiel hierfür ist die App für die Grundoberfläche. Es sind aber auch andere von Dritten oder selbst verfasste Anwendungen möglich.
Application Framework	Sind Funktionalitäten, die Android für jedes App bereitstellt unter anderem auch, das Apps angeben, welche Funktionen sie bereitstellen. Diese können dann von anderen Programmen genutzt werden.
Android Runtime	Ist im Grunde die Dalvik virtuell Maschine, die Java Programme auf Systemaufrufe abbildet und eine fast vollständige Implementierung der Java Grund Bibliotheken. Zu erwähnen ist hierbei, dass jede App eine eigene virtuelle Maschine bekommt, welches das System sehr stabil macht.
Libraries	Sind ausgewählte Laufzeitbibliotheken, wie von Linux Distributionen bekannt, jedoch für eingebettete Systeme optimiert. Es gibt auch direkt auf solche Systeme angepasste Bibliotheken. Die Funktionalitäten, die sie bereitstellen, werden durch das Application Framework zugänglich gemacht.
Linux Kernel	Hardware naher Teil des Systems, er lädt Treiber und verwaltet Ressourcen, wie Speicher oder Prozessorzeit.

2.3 Web Plattform

“The first Web site, createt by Tim Berners-Lee and Robert Cailliau at CERN (European Nuclear Research Center), consisted of a collection of documents from static content, encoded in HyperText Markup Language (HTML). Since then, theWeb has evolved from an enironment hosting simple and static hypermedia documents to an infrastructure to an infrastructure for the execution of complex applications.”Casteleyn u. a. [2, Introduction S. 1]

Der Autor gibt hier einen Überblick, wie sich Webseiten von inhaltlich festen Hypertext-

Ressourcen bis hin zu dynamisch erzeugten Inhalten entwickelten. Welche durch Programme bereitgestellt werden, die auch untereinander Daten austauschen können. Diese Anwendungen werden auch Web-Applicationen genannt. Sie werden laut [2, Introduction S.1-2] durch folgende Merkmale charakterisiert:

- Besserer Zugriff auf Informationen und Dienste: Im Vergleich zu lokalen Lösungen, die unflexibler und stationär gebunden sind.
- Dokument zentriertes Hypertext-Interface: Informationen werden über Strukturierte HTML-Dokumente angezeigt.
- Unterschiedliche Technologien zum Datenmanagement: Die Daten werden in verschiedenen Formaten, unter anderem XML, JSON, usw., anderen Anwendungen als Datenquelle zur Verfügung gestellt.
- Unterschiedliche Darstellungstechnologien und -engines: Es können verschiedene Ausgaben, zum Beispiel für mobile Geräte, angepasst werden.
- Komplexität der Architektur: Die großen Datenmengen werden ansprechent und abgestuft dargestellt.

Abschließend ist zu sagen, das Web Plattformen mit darauf Laufenden Web-Applikationen gut dazu geeignet sind, um Funktionalitäten einer großen Anwenderschaft zur Verfügung zu stellen. Da hier nicht nur Ressourcen wie Rechenzeit, sondern auch Speicherplatz für persönliche Daten zur Verfügung gestellt werden, ist eine erhöhte Sorgfalt gegenüber sicherheitsrelevanten Aspekten, wie in Kapitel 5 beschrieben, zu beachten.

2.4 Kommunikationsprotokolle

Die Arbeit behandelt nicht die unteren Protokollschichten, wie TCP/IP, da die darauf aufbauenden Protokolle Hyper Text Transfer Protokoll (HTTP) oder Hyper Text Transfer Protokoll Secured (HTTPS) auch den Datenverkehr abbilden können. HTTPS ist hierbei das HTTP welches über SSL/TLS verschlüsselt, und so gesichert wird. Im weiteren wird nun das HTTPS mit dem HTTP gleichgesetzt.[23, vgl.]

Das HTTP ist ein verbindungsloses Protokoll, das heißt, dass sämtliche Daten, die bei einem folgenden Aufruf vorliegen sollen, entweder über den Client zurückgegeben oder durch einen eineindeutigen Schlüssel auf dem Server gespeichert werden. Es ist ein Client Server Protokoll und arbeitet mit sogenannten Requests (dt. Anfragen) und Responses (dt. Antworten). Als solch ein Protokoll lädt es, meist textbasierte, Ressourcen von einem Webserver die durch eine Uniform Resource Locator (URL) meist mit einem Anhang, dem sogenannten Query String (dt. Abfrage Zeichenkette) welche die Ressource erst erschaffen oder verändern können.

Über dieses Protokoll kann man nun auch weitere Protokolle definieren, z.B. auf XML Basis.Diese Protokolle werden meist auf beiden Seiten validiert. Der Client beginnt und erstellt die erste Extensible Markup Language (XML) nach vorgegebenen Muster und sendet sie an den Server. Dieser validiert die Datei und verarbeitet sie weiter und sendet eine Antwort zurück, die wiederum vom Client validiert und verarbeitet wird. Mit diesen Mitteln kann auch ein verbindungsorientiertes Protokoll erstellt werden. Wie im Rahmen dieser Arbeit entwickelt wurde. Näheres dazu im Abschnitt 6.6.

2.5 Installation und Konfiguration der Entwicklungs und Produktivsysteme

Sofern noch nicht geschehen müssen folgende Programme (vgl. Tabelle: 2.1) installiert werden. Für die Betriebssysteme gibt es natürlich auch andere Möglichkeiten, allerdings werden sie in der Arbeit, mit Verweis auf die jeweilige Fachliteratur, nicht behandelt. Warum hier die Entscheidung auf Systeme rund um den Linux Kernel fiel, wird weiter unten erklärt. Die Systeme sind unterteilt in das Entwicklungssystem und zwei Produktivsysteme. Letzteres soll hier ein entfernter, gemieteter Linux Server ohne grafische Oberfläche sein. Die Entwicklungssystem haben im Gegensatz dazu eine grafischen Oberfläche.

Gemeinsamkeit bei beiden Systemarten ist die Installation einer frei wählbaren Linux Distribution. Im Rahmen der Arbeit fiel die Wahl auf Fedora 15 (Fedora) und Ubuntu 11.04 (Ubuntu) für die Entwicklungssysteme und openSUSE 10.3 (openSUSE) für das Produktivsystem. Das Betriebssystem der Produktivumgebung war schon länger für andere Projekte der Firma in Gebrauch, deshalb wurde es für dieses Projekt übernommen. Das Entwicklungssystem in der Ubuntu Variante wurde gewählt, da schon im Studium mit diesem System gearbeitet wurde. Fedora wurde genutzt, da es viele Neuerungen wie die Ersetzung des SystemV init Mechanismus mit Upstart Ansatz und später mit Systemd [vgl. 21] und neuere Softwareversionen nutzt [vgl. 22], um mögliche Einflüsse auf das zu erstellende Programm absehen zu können.

Programme wie der Webserver mit PHP Unterstützung und die SQL Datenbank sind nicht Teil des Betriebssystem und müssen nachinstalliert werden. Dafür sind natürlich die passenden Rechte nötig, dies kann über Mechanismen wie su oder den Login als root[1], sowie für einzelne Befehle *sudo* erreicht werden. Wie sie im einzelnen Konfiguriert werden, ist nicht Gegenstand dieser Arbeit, dafür kann Fachliteratur zu Rate gezogen werden. Die Kommandozeilenbefehle an denen Super User Rechte nötig sind, werden im weiteren durch ein # gekennzeichnet, die bei denen Nutzerrechte ausreichen mit $ nach der üblichen UNIX Terminal Symbolik. Die Installation ist für die einzelnen Distributionen folgendermaßen zu bewerkstelligen:

[1] in einigen Distributionen aus Sicherheitsgründen ohne entsprechende Konfiguration nicht möglich

Fedora: Es wird standardmäßig yellowdog updater modified (yum) genutzt, um Programme zu installieren. Mit *# yum groupinstall 'Web Server'* werden außer dem Apache Server gleich einige Unterstützungsprogramme wie PHP oder perl mit installiert. Um nun den Server zu starten muss, folgender Befehl eingegeben werden: *# service httpd start*. Um den Server Prozess schon bei dem Systemstart automatisch auszuführen ist folgender Befehl notwendig: *# chkconfig httpd on*.[vgl. 20]

Um den MySQL Server zu installieren, ist nur der Befehl *# yum install mysqlserver* erforderlich. Wie der Apache Webserver muss auch dieser Server gestartet werden. Dazu sind die gleichen Befehle notwendig, allerdings mit der Variation, das httpd durch mysqld ersetzt wird.

Ubuntu: In Ubuntu wird standardmäßig das advanced packaging toolkit (apt) eingesetzt. Um dabei einen Linux Apache MySQL PHP/Perl/Python (LAMP) Stack zu installieren, sind folgende Schritte nötig. Zuerst die Installation von tasksel ein Programm um Programmgruppen, die auf Aufgaben ausgerichtet sind zu installieren [vgl. 4]. Dann die Installation des LAMP Stacks. Folgende Befehle sind auszuführen:

- um tasksel zu installieren: *# apt-get install tasksel*
- um den LAMP-Stack zu installieren: *# tasksel install lamp-server*

Hier werden der mysql sowie der apache Services beim Systemstart gestartet und sollten auch sofort nach Installation laufen.

openSUSE: Auf dem angemieteten Produktivsystem ist normalerweise schon ein LAMP Stack installiert. Da diese Systeme meist über ein Webfrontend konfigurierbar sind. Sollte dies nicht der Fall sein, so ist ein Zugang per secure shell (SSH) möglich. Die Installation erfolgt hier über yast. Folgendermaßen wird ein LAMP Stack installiert: *# yast2-install apache2 php5 php5-mysql apache2-mod_php5 mysql mysql-tools*. Die Dienste müssen noch gestartet werden. Dies geschieht so: *# rcapache2 start* für den Webserver und *# rcmysql start* für den MySQL Server. Hier werden die fertigen Dateien des entwickelten Programms in den Ordner /srv/www/vhosts/virtual_host/httpsdocs kopiert. Es muss noch die vom Entwicklungssystem exportierte Datenbankstruktur importiert werden, damit es lauffähig wird. Dazu ist ein Programme nötig, welches meist schon installiert ist, es nennt sich phpMyAdmin mit dem man die Datenbank administrieren kann.

Um zu testen, ob die Installation und das Starten erfolgreich war, kann in einem beliebigen auf dem System installierten Browser *http://localhost* eingegeben werden, was dann nicht zu Fehlern führen sollte, Ähnliches zur Folge hat. Bei dem Produktivsys-

tem ist dies auf einem anderen System durchzuführen, dabei ist auch die Zeichenkette localhost durch die Server URL zu ersetzen.

Es muss noch ein MySQL root Nutzer erstellt werden. Das geschieht mit folgendem Befehl: *# mysqladmin -u root -p rootpassword*. Letztendlich muss nur noch die Struktur der Datenbank erstellt, sowie ein Nutzer, der jeweiligen Rechte auf die Datenbank hat. Dazu wird einfach das MySQL Workbench Programm installiert, darin die erforderliche Struktur (siehe 6.1) und ein dazugehöriger Nutzer erstellt und an die Datenbank gesendet.

Um mit den Entwicklungsysteme arbeiten zu können, benötigt man noch eine Entwicklungsumgebung. Sie muss PHP, Hypertext Markup Language (HTML) und Cascading Style Sheets (CSS) unterstützen, so wie ein Plugin für das Android SDK bereitstellen. Im Zuge der Entwicklung wurde hierfür NetBeans in einer Version >=7.0 eingesetzt. Sie ist unter *http://netbeans.org/downloads/index.html* in unterschiedlichen Versionen downloadbar. Da es bei den meisten Distributionen noch kein Paket für diese gibt, kann man sie einfach als .bin-Datei herunterladen. Diese Datei muss nur wie folgt in einer Konsole ausgeführt werden: *# /path/to/file/name.bin*. Zu beachten ist hier, dass sie als root ausgeführt wird. Im Installationsprozess werden die Features aufgeführt, hier können die zu unterstützenden Programmiersprachen ausgewählt werden. Es fehlt nur noch das Android Plugin. Man öffnet den NetBeans Menüpunkt Tools->Plugins und dan Settings fügt ein neues Update Center hinzu mit der URL *http://kenai.com/projects/nbandroid/downloads/download/updatecenter/updates.xml*.(vgl. Abbildung 2.3). Danach kann man in Available Plugins nach Android suchen und das Android Plugin installieren.

Um Programme für Android Systeme zu entwickeln benötigt man noch das dazugehörige SDK, es ist über die Website *http://developer.android.com/sdk/index.html* erhältlich. Nachdem es gedownloaded ist, kann es an beliebiger Stelle entpackt werden. Im Zuge der Entwicklung ist es nötig mit dem Programm *<SDK Verzeichnis>/tools/android* ein Testsystem, ein sogenanntes Android Virtual Device, zu erstellen.(vgl. Abbildung 2.4) In dieser Arbeit wird Android 2.2 als Testsystem verwendet. Alternativ ist auch die Verwendung eines Androidgerätes ab Version 2.2 möglich. Dieses muss dazu per Universal Serail Bus (USB) angeschlossen sein und die Option USB Debugging aktiviert haben. Das Testsystem wird mit Tastatur am Rand und Touchscreen emuliert, der Touchscreen wird dabei per Maus bedient und alle Eingaben können auch per Tastatur erfolgen.(vgl. Abbildung 2.5)

Abschließend sind Einstellungen in den Entwicklungssystemen vorzunehmen. Das NetBeans Android Plugin benötigt den Ort an dem das SDK entpackt wurde. Das geht über den Menüpunkt Tools->Options->Miscelaneos->Android. Weiter ist es zweckmäßig die Konfiguration des Apache Webserver so abzuändern, das er auf die Dateien der NetBeans Projekte Zugriff hat. In Fedora kann man das erreichen, indem man eine Datei

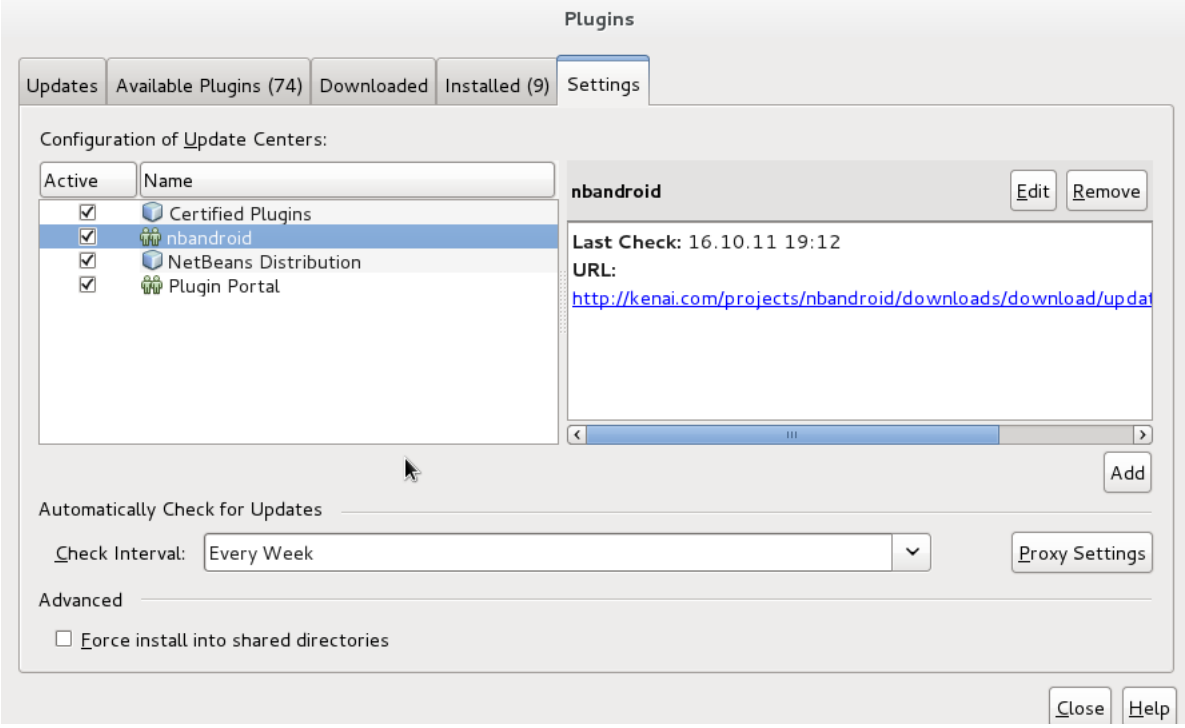

Abbildung 2.3: NetBeans Plugin Konfiguration

im Ordner */etc/httpd/conf.d* erstellt. Diese Datei muss mit root Rechten erstellt werden und folgenden Aufbau haben:

Listing 2.1: Konfiguration des Webservers in den Entwicklungssystemen

```
#/etc/httpd/conf.d/dev.conf
Alias /dev /path/to/NetBeansProjects
<Directory /path/to/NetBeansProjects>
        Order Deny, Allow
        Deny from All
        Allow from 127.0.0.1
        Allow from ::1
</Directory>
```

Allerdings ist hier auch Security Enhanced Linux (SELinux) aktiv was einen weiteren Befehl nötig macht: *# chcon -R reference=/var/www/html /path/to/NetBeansProjects*. Er setzt die Regeln(vgl. Abschnitt 5.1) des default Directorys auch für den Ordner in dem die NetBeans Projekte liegen. Die Zeichenkette */path/to/NetBeansProjects* ist bei der Datei sowie dem Befehl wörtlich zu nehmen und mit dem Pfad zu dem Ordner in dem die NetBeans Projekte liegen zu ersetzen. Zuletzt müssen noch die Rechte für die Dateien verändert werden dazu führt man in dem NetBeansProjects Verzeichnis den Befehl *$chmod a+xr <Web-Projekt>* aus. Die Zeichenkette Web-Projekt ist dabei mit dem Namen derWeb Applikation zu ersetzen. Damit hat der Apache Prozess Lese- und Ausführungsrechte für die Dateien in diesem Ordner und im Browser sollte dann nach Eingabe von *http://localhost/dev/<Web-Projekt>* das gewünschte Projekt angezeigt werden.

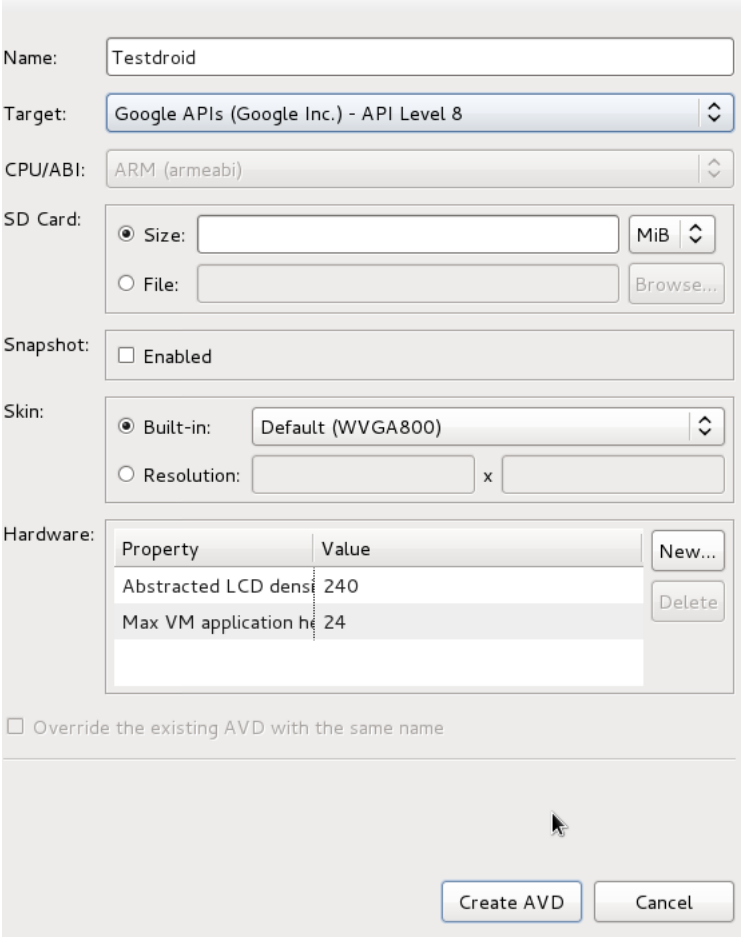

Abbildung 2.4: Erstellen des Android Virtual Device

In einer Ubuntu Installation kann der SELinux-Schritt entfallen. Außerdem liegen hier die Webserver Konfigurationsdateien in dem Verzeichnis */etc/apache2/conf.d*. Die restlichen Einstellungen können für dieses System übernommen werden.

Tabelle 2.2: Programme die genutzt werden

Programm	System	Bemerkung
Ubuntu 11.04 (x86-64)	Entwicklungssystem	Betriebssystem für das erste Entwicklungssystem
Fedora 15 (x86-64)	Entwicklungssystem	Betriebssystem für das zweite Entwicklungssystem
openSuse 10.3 (x86-64)	Produktivsystem	Betriebssystem für das Produktivsystem
Apache (>=2.2)	Entwicklungssysteme, Produktivsystem	Webserver der auf lokale oder externe HTTP Anfragen reagiert.
MySQL (>=5)	Entwicklungssysteme, Produktivsystem	SQL Server der auf beiden Systemen nur lokale Anfragen entgegennimt.
PHP (>=5) apache_php module	Entwicklungssysteme, Produktivsystem	Serverseitiger PHP-Scriptinterpreter zum erstellen dynamischer Webseiten. Diese Version muss genutzt werden, da durch sie erst viele Techniken der Objektorientierung in PHP Einzug hielt.[vgl. 18]
NetBeans 7.1	Entwicklungssysteme	Entwicklungsumgebung, die eine Vielzahl von Programmiersprachen unterstützt unter anderem JAVA und PHP.
Android Plugin für NetBeans	Entwicklungssysteme	Erweitert NetBeans um die Fähigkeit Android Programme zu erstellen und testen.
Android SDK	Entwicklungssysteme	Zusammenstellung von Android Entwicklungswerkzeugen, unter anderem ein Geräteemulator und Steuerprogramme.

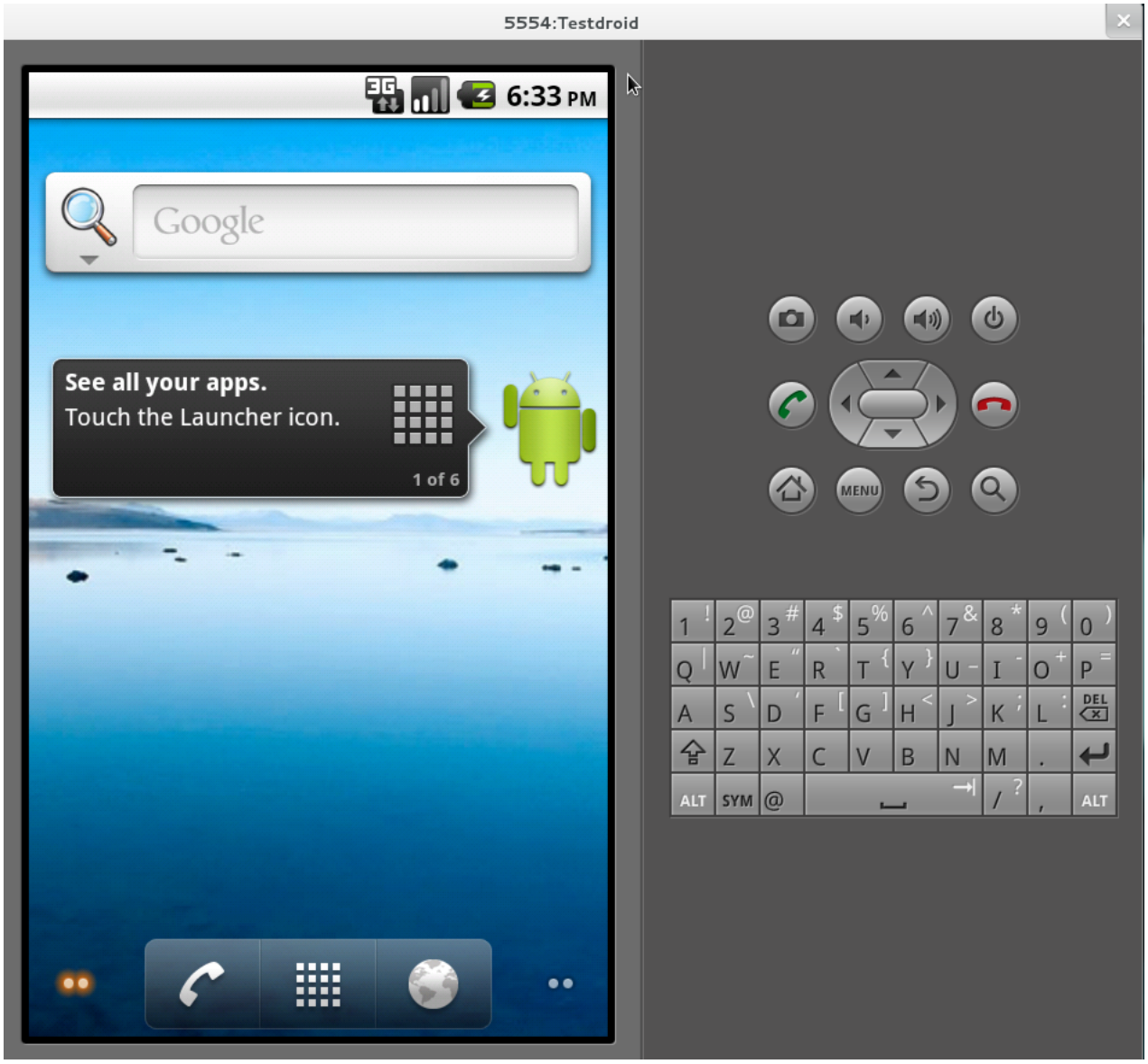

Abbildung 2.5: NetBeans Plugin Konfiguration

3 Grundabläufe vor Systemeinführung

Dieses Kapitel beleuchtet die vereinfachten Geschäftsabläufe vor dem Programmeinsatz. Die Darstellungen stützen sich auf einen internen Bericht der Mitarbeiter aus Lettland. Sie werden in folgenden Abschnitten stark vereinfacht wiedergegeben.

3.1 Adressen Sammeln

Die Mitarbeiter suchen Orte auf, an denen eine große Anzahl von Interessenten vermutet wird. Dazu werden Lokalitäten aufgesucht, an denen die Zielgruppen vermehrt anzutreffen sind. Es sind zum Beispiel ausgesuchte Messen, größere Kaufhallen oder belebte Plätze. Dort werden Passanten angesprochen und meist über eine Gewinnspielteilnahme deren Adressen gesammelt. Dies geschieht in Papierform. Am Ende dieser Sammelaktionen werden die Adressen dann händisch oder halbautomatisch auf Vollständigkeit und Dubletten geprüft. Wenn in der Niederlassung technische Ressourcen vorhanden sind, werden die Daten stationär eingepflegt. Ziel ist es einen Stamm potentieller Kunden zu generieren, der zu Produktpräsentationen oder Telefonverkäufen herangezogen werden kann.

3.2 Auswahl der Interessenten zur Kontaktaufnahme

Der Telefonistenleiter selektiert die Daten Adressenkarten nach den Prämissen, Alter, Familienstand usw. die dann in Listen zusammengefasst weitergegeben werden. Datensätze die aufgrund von oben genannten Filterkriterien ausgefiltert wurden, verbleiben in der Datenbank, um Auswertungen durchführen zu können. Die Listen sind außerdem Produktgruppen und Wohngegenden zugeordnet, sie sind gleichzeitig Veranstaltungen fest zugeordnet. Diese Listen werden nun den operativen Telefonisten übergeben.

3.3 Planen der Veranstaltungsorte

Vor der Terminierung werden Räumlichkeiten in Cafés und Restaurants gebucht. Dies geschieht auf der Grundlage der Veranstaltungsplanung. Um den Aufenthalt der Gäste in den verschiedenen Produktvorführungen angenehmer zu gestalten wird nach betrieblichen Vorgaben die Bewirtung geregelt.

3.4 Terminierung

Der Telefonist informiert den Interessenten über den Gewinn der Verlosung informiert und schlägt den Besuch einer Produktpräsentation vor. Sie sind zeitlich ungefähr eine Woche von diesem Gespräch entfernt. Außerdem wird nach möglichen weiteren Interessenten gefragt. Es werden zwei weitere Gespräche geführt um eventuelle Unpässlichkeiten seitens der Gäste auszuschließen und sie an den Termin zu erinnern. Das letztendliche Ziel des Telefonisten ist 8 Gäste vorzugsweise aus vier Paaren bestehend auf einen Termin zu buchen. Dazu sind nicht immer 3 Telefonate möglich, zum Beispiel wenn eine Veranstaltung durch Absprung einzelner Gäste drei Tage vor Termin unterbesetzt ist.

4 Sytemgestützte Abläufe

In diesem Kapitel werden die durch das Programm optimierten Abläufe widergespiegelt. Dazu werden Use Case Diagramme herangezogen, Abläufe für die einzelnen Use Cases werden in Abschnitt 6.2 angegeben und erläutert.

4.1 Adressen Sammeln

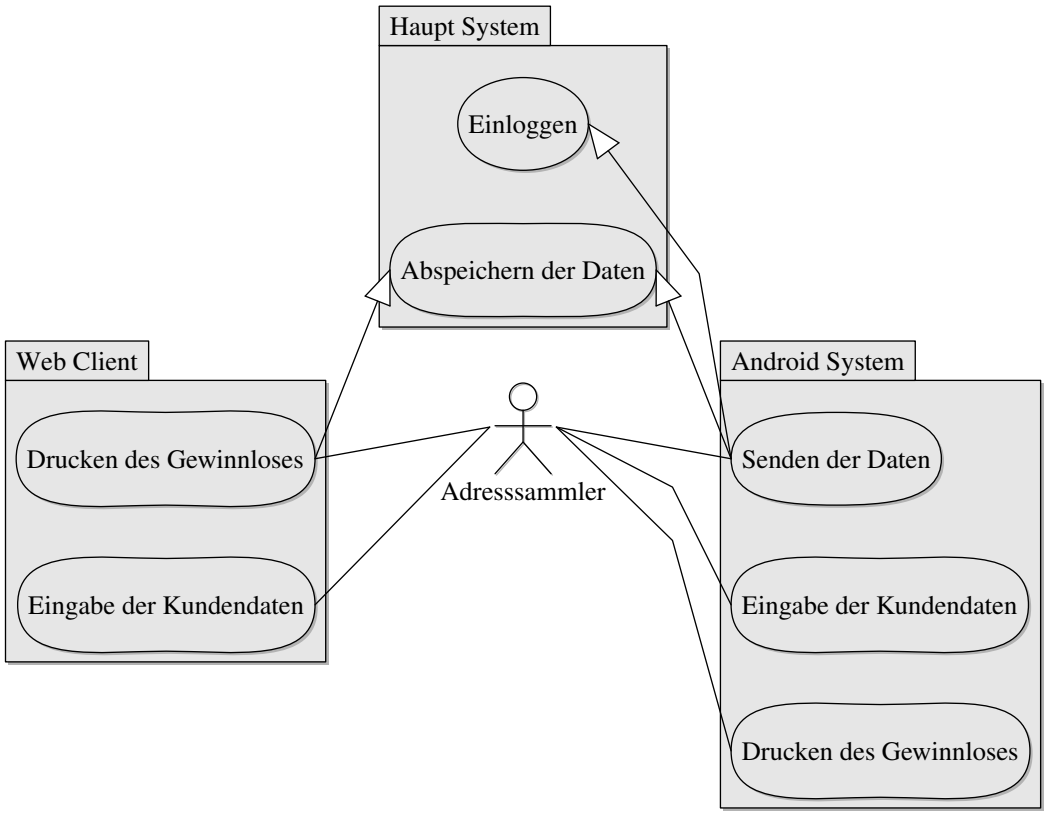

Abbildung 4.1: Use Case Diagramm für das Adressen Sammeln

Der Adresssammler kann hier mit zwei, größtenteils voneinander unabhängigen, Systemen interagieren. Zum einen gibt es das Hauptsystem, welches mittels einer Weboberfläche angesprochen wird. Hier können alle Kundendaten direkt in die Datenbank eingepflegt werden. Diese Daten liegen dann sofort für die Weiterverarbeitung vor. Das zweite System stützt sich auf Android Plattformen. Hier können alle Kundendaten lokal auf dem jeweiligen Gerät eingepflegt werden. Es ist möglich Gewinnlose mit einer eindeutigen ID zu drucken, welche später auch zu Verlosungen herangezogen werden können. Der Datenbestand wird dann durch Nutzerinteraktion auf das Hauptsystem übertragen. Im Fall von Übertragungsfehlern werden die Daten in der Datenbank des mobilen Gerätes zurückgehalten und es wird eine Fehlermeldung erzeugt. Die Übertragung kann

später erneut durchgeführt werden. Es ist angedacht, dass jeder Adresssammler den Upload der Daten, es handelt sich hierbei um 100-200 Datensätze, täglich durchführt. Das Hauptsystem arbeitet die Daten dann automatisch in den Datenbestand ein.

4.2 Planen von Veranstaltungsorten

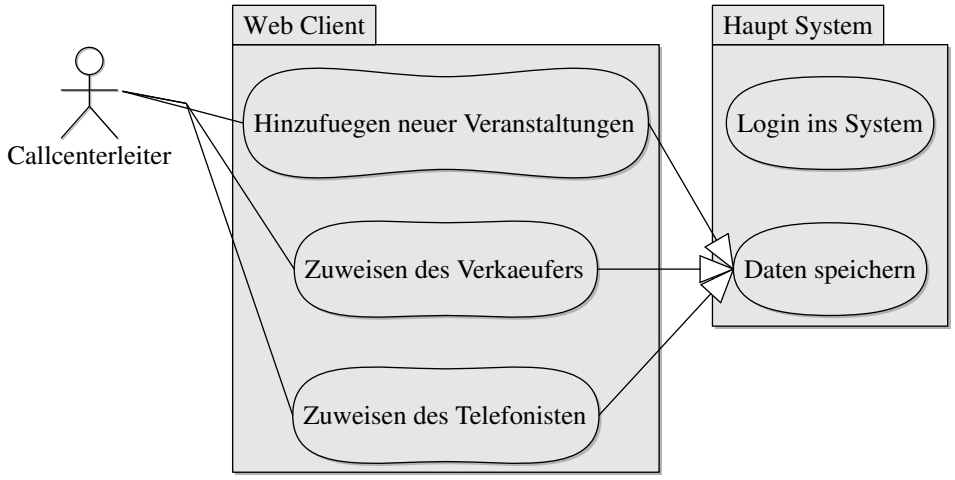

Abbildung 4.2: Use Case Diagramm für Veranstaltungsplanung

Der Veranstaltungsort wird durch den Aktor Telefonistenleiter erstellt. Diese Aufgabe ist monatlich durchzuführen. Es werden jedoch wöchentlich Verfeinerungen vorgenommen um unvorhergesehene Probleme zu berücksichtigen. Die Telefonisten werden den zu betreuenden Veranstaltungsorten zugeordnet.

4.3 Auswahl der Kunden zur Kontaktaufnahme

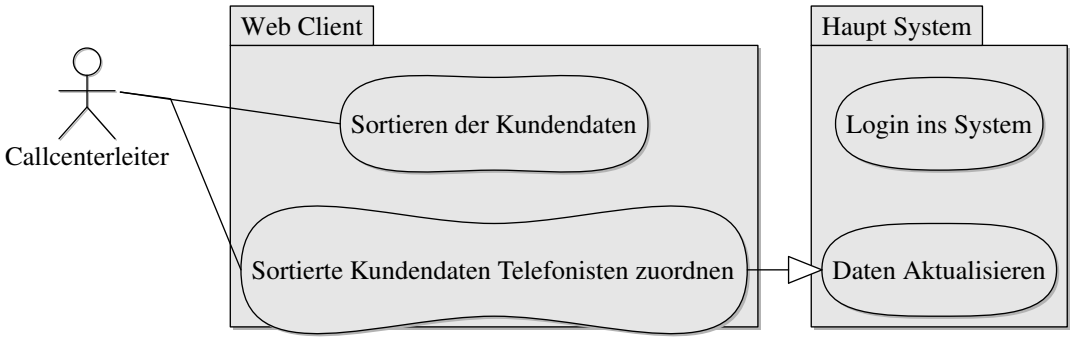

Abbildung 4.3: Use Case Diagramm für Kundenauswahl

Für die Kontaktaufnahme mit den Kunden ist nur das Hauptsystem vorgesehen. Es wird von dem Telefonistenleiter bedient. Alle Interaktionen werden über eine Weboberfläche durchgeführt. Dem Leiter sollen hierbei Filtermethoden helfen, dem Telefonisten Kunden zuzuteilen.

4.3.1 Terminierung

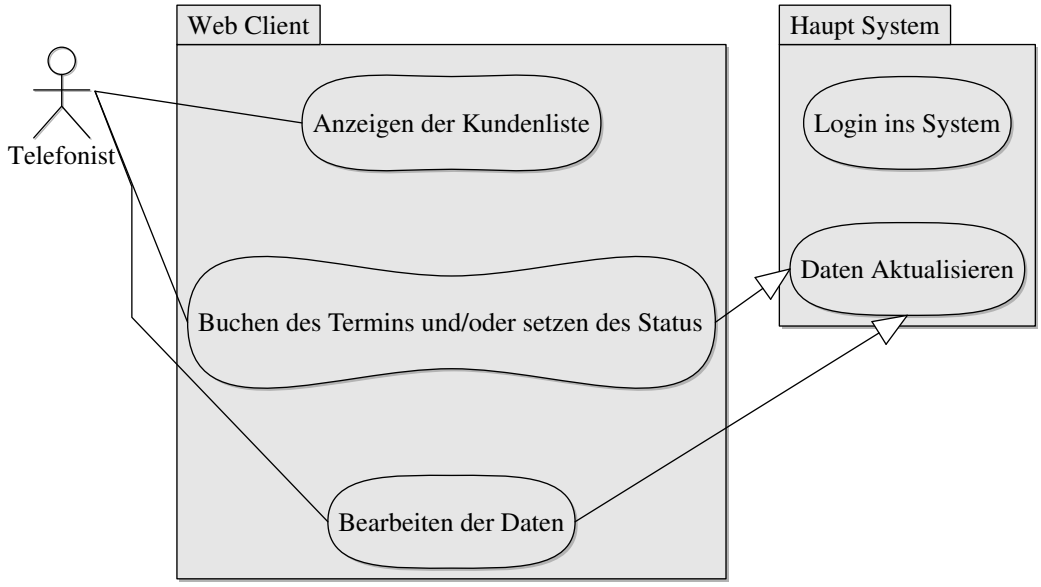

Abbildung 4.4: Use Case Diagramm für Terminierung

Dem Telefonisten ist es nun möglich seine jeweilige Kundeliste abzutelefonieren und damit für die verschiedenen Produktvorführungen zu werben. Sie buchen auch die Plätze für die Veranstaltungen. Die Farbgebung der einzelnen Veranstaltungstermine orientiert sich am Ampelsystem. Die ungefüllten Veranstaltungstermine sind rot gekennzeichnet, wenn sie gefüllt sind werden sie grün gekennzeichnet. Wenn an einem Termin nur noch 80 Prozent zur Verfügung stehen, soll dieser gelb hinterlegt sein. Es sind drei Telefonate mit den Kunden bis zur Verkaufsveranstaltung vorgesehen. Nach dem Ersten sollte feststehen, ob der Kunde die Einladung annimmt. Daraufhin wird automatisch eine schriftliche Einladung mit Anfahrtsbeschreibung und dem Namen des persönlichen Ansprechpartners gedruckt und versandt. Wenn der Kunde absagt, wird er für eine bestimmte Zeit gesperrt. Im zweiten Gespräch wird zu erst noch einmal die Bestätigung des Erscheinens abgefragt und dann die Anzahl der weiteren Gäste und gegebenenfalls deren Namen ermittelt. Das letzte Gespräch soll nochmal der Erinnerung dienen und eventuelle Änderungen der Personenzahl ergeben, um dies für die Veranstaltungsplanung zu berücksichtigen. Diese Telefonate werden mithilfe der Status angezeigt und später und vom System automatisch wieder zur Vorlage gebracht.

5 Sicherheitsaspekte

Da hier Kundendaten, die für Unternehmen von großer wirtschaftlicher Bedeutung sind, gespeichert werden. Ist die Anforderung an die Sicherheit sehr hoch einzustufen. Das gilt sowohl für Außenstehende, die gar keinen Zugriff auf das System haben sollen, als auch für Mitarbeiter des Betriebes. Letztere sollen nur Zugriff auf Daten haben, die explizit für sie freigegeben wurden. In diesem Kapitel werden sinnvolle Sicherheitseinstellungen, des Systems vom Betriebssystem über den Webserver bis hin zu den Skript-Sprachen für dynamische Inhalte und Client-System Android vorgestellt.

5.1 Betriebssystem des Servers

Der Administrator sollte sich regelmäßig über Schwachstellen informieren, um diese zeitnah zu beseitigen. Weiter sollte darauf geachtet werden, alle nicht nötigen Komunikationskanäle zu schließen.

Hier wird speziell auf Linuxbasierte Server eingegangen. Zuerst werden die Ports welche nicht benötigt werden gesperrt, dies wird durch Kernel interne Mechanismen, *iptables* realisiert.[16] Eine für den reinen Webserverbetrieb über sichere Verbindung zugeschnittene Konfiguration würde so aussehen[2]:

Tabelle 5.1: Übersicht über die Konfiguration von IP-Tables[3]

Line	Rule	Funktion
1	*-P INPUT DROP*	Alle einkommenten Packete werden verworfen
2	*-P OUTPUT DROP*	Alle ausgehenden Packete werden verworfen
3	*-P FORWARD DROP*	Alle Packete werden für andere Rechner werden verworfen
4	*-A OUTPUT -p tcp –dport 443 -j ACCEPT*	Hier werden alle Packete, die auf port 443 (HTTPS) ankommen duchgelassen
5	*-A OUTPUT -p tcp –dport 21 -j ACCEPT*	Hier werden alle Packete, die auf port 22 (SSH und SFTP) ankommen duchgelassen

Um die Sicherheit weiter zu erhöhen, wurde das Projekt SELinux, welches die Flux Ad-

[2] In Fedora 15 */etc/sysconfig/iptables*, Ubuntu 11.04 */etc/iptables.up.rules*

vanced Security Kernel (FLASK) Architektur implementiert, erstellt.[vgl. 14] Die Installation erfolgt auf den in Abschnitt 2.5 Seite 7 eingeführten Systemen folgendermaßen:

- Ubuntu: *# apt-get install selinux*
- Fedora: Hier wird SELinux schon mit dem Betriebssystem installiert.
- openSuse: Wird erst ab Version 11.1 unterstützt, ist darin vorinstalliert und muss nur wie folgt aktiviert werden.

Das Sicherheitssystem kann wie folgt de-/aktiviert werden. Die Datei /etc/selinux/config muss Konfiguriert werden, dazu ist die Option SELINUX auf gewünschten Wert zu setzen[vgl. 5]:

- auf disabled für deaktiviert
- auf enforcing für das erzwingen der Regeln
- auf permissive um vor Verletzungen der Regeln zu warnen

Es bringt erweiterte Rechte für Programme und Nutzer. Diese Restriktionen sind nicht nur auf das Dateisystem begrenzt, sondern auch für Ressourcen wie das Netzwerk anwendbar. Sie werden über ein Regelwerk, die sogenannten Policys festgelegt [vgl. 9]. An der Einrichtung dieser Technik, die von den Distributionen vorgegeben ist, ist meist keine Änderung nötig. Die Konfiguration beschränkt Programme nur auf für sie vorgesehene Dateien/Verzeichnisse/Ressourcen Zugriff haben. Wie eine angepasste Konfiguration zu erstellen ist kann in den Dokumentationen der jeweiligen Distribution oder in der Fachliteratur nachgelesen werden.

Es ist ratsam ein Intrussion Detection System zu installieren. Diese Systeme analysieren den Datenverkehr und informieren den Administrator über ungewöhnliche Ereignisse. Zu nennen ist in diesem Sektor Suricata, es wird über ein Regelwerk konfiguriert[vgl. 11]. Hier sind die mit installierten Regeln auch völlig ausreichend. Der Administrator kann dann auf geeignete Weise auf diese Benachrichtigungen reagieren. Die Installation er- folgt über die für die Distribution üblichen Programme mit dem Paketnamen suricata.

Um kompromittierten Konfigurationsdateien vorzubeugen, ist es möglich diese unter eine Versionsverwaltung zu stellen. Eine mögliche Wahl wäre hierbei **Git**. So werden die Veränderungen über längere Zeiträume protokolliert. Dies ist nicht nur ein sicherheitsrelevanter Abschnitt, er kann auch helfen, die Downzeiten zu vermindern, indem die gesamte Konfiguration ausgecheckt und auf einen anderen Server übertragen werden kann. Ein Programm, welches diesen Ansatz verfolgt ist Etckeeper [vgl. 13]. Die Konfiguration wird in der Datei */etc/etckeeper/etckeeper.conf* durchgeführt und die einzelnen

Punkte werden darin erklärt.

Für Neuentwicklungen in diesem Sektor sollte man dennoch immer offen sein, jedoch zwischen Performance und Sicherheit abwägen.

5.2 Webserver und angegliederte Programme

In diesen Rubrik fallen:

- Das Webserver Programm selbst, hier der Apache Webserver, der am häufigsten eingesetzte Webserver[vgl. 15].
- Das dahinter liegende Datenbank-Programm, hier der Mysql-Server.
- Der Interpreter für die Server-seitige Skriptsprache, hier speziell der PHP-Interpreter.
- Sekundär Programme, die nicht an der Primär-Funktion des Servers beteiligt sind.

Diese Applikationen verarbeiten oder speichern Daten von Nutzern. Bei ihnen kann Mithilfe bestimmter Eingaben oder Eingabemengen ein vom Entwickler ungewolltes Verhalten provoziert werden.

Apache Webserver: Hier empfiehlt es sich den Server unter einem eigenen Nutzerkonto mit wenigen Privilegien laufen zu lassen.[3] Nachdem er nun nur auf einige Verzeichnisse/Dateien Zugriff hat, kann man noch den Zugriff auf Unterverzeichnisse oder Dateien mittels *.htaccess* einschränken. Er muss gegen Denial of Service (DOS)-Attacken geschützt werden. Das sind Attacken wo der Serverprozess so überlastet wird, das er jeden weiteren Verbindungsaufbau ablehnt, zu schützen. Um dies zu erreichen, kann man erst einmal die Timeoutzeit verringern. Sie ist die Zeit, die der Server für den Wiederaufbau der Verbindung wartet zu verringern. Um den gesamten Datenverkehr zum Beispiel vor Man in the Middle Angriffe (Angriffe wo ein System sich zwischen Server und Client schiebt und damit den ganzen Datenverkehr Protokolliert) zu schützen, sollte man gerade bei sensiblen persönlichen Daten eine verschlüsselte Übertragung, z.B. mit SSL3 oder höher in Betracht ziehen, wie in [8, S. 64ff] beschrieben.

MySQL Server: MySQL nutzt Access Control Lists (ACLs) zur Zugriffskontrolle. Es ist so aufgebaut, das auch die SQL Benutzerkonten, sowie das zugehörige Passwort, als Hashwert, in einer Tabelle *user* gespeichert sind. Diese Tabelle sollte nur für root Konten, das sind auch hier die Datenbankadministratoren, einsehbar und veränderbar sein. Diese Zugänge sollten natürlich immer über ein Passwort

[3] ist bei den meisten Distributionen schon nach der Installation so, z.B.:*www:www*

geschützt sein. Da der Server hier nur lokal eingesetzt wird, sollte auch nur der Zugriff über ein lokales Socket gestattet werden.

Schritte um dies sicherzustellen sind:

- Auf der Serverkonsole *mysql -u root* einzugeben, sollte dieser Aufruf, ohne Passworteingabe, zu einer MySQL Shell führen, ist kein SQL-Passwort gesetzt.

- Die SQL Anweisung *SHOW GRANTS;* zeigt welche Nutzer mit welchen Rechten auf welche Tabellen Zugriff haben. Hier kann man aufgrund der Daten abwägen, wie viel Rechte für jeden einzelnen Nutzer notwendig sind. Die Rechte können dann mit *REVOKE* gesetzt werden.

- In der Konfigurationsdatei[4] sollte ein fester Nutzer eingestellt werden. Die Loopback-Adresse sollte die einzige Adresse sein an die der MySQL Server gebunden ist. Eine Beispielkonfiguration könnte so aussehen:

```
[mysqld]
bind-address=127.0.0.1
user=mysql
```

PHP Interpreter: Der Interpreter wird hier als Apache Modul eingebunden. Als solches übernimmt es den Nutzer und die Gruppe des Webservers. Die Skriptdateien müssen somit für diesen Nutzer lesbar sein. Sollte es möglich sein, das der Nutzer Dateien hoch laden kann, sollte man diese nur in vordefinierten Verzeichnissen speichern, wo der Interpreter Schreibrechte hat. Auch nur für einzelne durch PHP änderbare Dateien sollten Schreibrechte erhalten. Die restlichen Dateien und Verzeichnisse sind für Veränderungen zu sperren. Sekundär Programme: Programme die insbesondere auch eine Verbindung nach außen darstellen, sind oft auch ein Risiko für die Sicherheit des Systems.

- File Transfer Protocol (FTP): Hier sollte wie auch bei HTTP das abgesicherte Protokoll SSH File Transfer Protocol (SFTP) genutzt werden. Es werden unterschiedliche Nutzer für die verschiedenen Webordner angelegt. Keinesfalls sollte er alle Verzeichnisse durchsuchen und gar jede Datei einsehen können, da so die genutzten Programme und auch Schwachstellen in der Konfiguration ausgespäht werden können. Die Konfiguration kann in der Fachliteratur nachgelesen werden.

- SSH: Ist ein Standard, welcher auf Servern zur Fernwartung eingesetzt wird. Es ist sicherer als das veraltete Telnet, welches alle Daten unverschlüsselt überträgt. Man kann aber auch hier etwas mehr Sicherheit erzwingen. So

[4] Fedora 15: */etc/my.cnf*, Ubuntu 10.04: */etc/mysql/my.cnf*

kann man die Anmeldung des root Nutzers unterbinden. Der Nutzer kann dann, wenn die Programme dafür konfiguriert sind, entweder sudo für einen einzelnen Befehl oder su für eine root Sitzung starten. Auch hier ist steht zum nachlesen der Konfiguration Fachliteratur bereit.

5.3 Scripte, die auf dem Webserver laufen

Hier gibt es eine klare Grundregel, man sollte Nutzereingaben nicht trauen. Das heißt, dass man keine Eingaben ungeprüft lassen soll. Sicherheitslücken welche entstehen, sind unter anderem Cross Site Scripting (XSS) und Structured Query Language (SQL) Injections. XSS ist dabei ein Angriff in dem Inhalte wie Verlinkungen, bis hin zu eingebettetem Javascript oder CSS in einer Internetseite eingefügt werden, um deren Funktion oder Aussehen zu ändern. Dies kann unter Umständen auch rechtliche Konsequenzen nach sich ziehen, wenn rechtlich unzulässige Inhalte eingebunden werden oder darauf verlinkt wird. Um dem entgegenzuwirken sollte man entweder HTML- Tags ganz verbieten oder ausgewählte Tags durch andere Zeichenketten ersetzen. SQL Injections sind nicht weniger schadhaft. Der Ablauf für diese Schwachstelle ist folgendermaßen:

1. Der Angreifer gibt SQL Code in ein Formular ein.

2. Er sendet es ab.

3. Er prüft die Reaktion der Webapplikation.

So kann er sich zum Beispiel ein Bild über den Aufbau der Datenbank machen oder gar für ihn sonst unzugängliche Daten auslesen. Hier gibt es mehrere Möglichkeiten diese Lücke vermindern zu können. Zum einen können Zeichen, die eine Sonderbedeutung haben, mit Hilfe von PHP Funktionen Escaped werden, das heißt, diese Zeichen Programmiertechnisch zu entwerten. Auf der anderen Seite kann man es mit Hilfe von prepared Statements, die Daten mit Sonderzeichen ohne schädliche Seiteneffekte verarbeiten. Die Überprüfung übernimmt hierbei der Datenbanktreiber. Er ist die Verbindung zwischen Programm und Datenbank und kümmert sich um die Kommunikation zwischen beiden [vgl. 19].

6 Umsetzung

Das Kapitel enthält eine Dokumentation der Umsetzung. Sie geht von den Usecases über die Datenbank bis hin zu Schwerpunkten der Implementierung.

6.1 Bestehende Lösungen

Das System hat den Fokus, auf den Ablauf von Kommunikation und Handelsbeziehungen zwischen Kunden und Firmen. Ziel ist die Bedienung für viele Mitarbeiter bereitzustellen. Alle Interaktionen sollen protokolliert werden. In diesem Zusammenhang ist es wichtig das jeder Mitarbeiter, der das System nutzt, ein eigenes Nutzerkonto hat. Es handelt sich dabei pro Filiale um rund 65 Mitarbeiter. Andere Systeme wurden in Betracht gezogen. Es existiert jedoch kein Ansatz welcher die Sammlung von Kundendaten in der priorisierten Form verfolgt. Als Resultat wurde dieses Programm entworfen, da es einen erheblichen finanziellen Mehraufwand bedeutet, andere Systeme anzupassen.

6.2 Use Cases

Hier werden die Abläufe der einzelnen Use Cases erklärt, um einen Überblick für die zu implementierenden Strukturen zu geben.

Use Case Einloggen: Das Einloggen ins Hauptsystem geschieht über eine einfache Texteingabe im Web Interface. Die Logindaten werden an den Server übermittelt, der die Daten prüft. Gleichzeitig werden die Rechte des Nutzers und dessen ID in einer statischen Variable speichert. Hier wird auch die Sprachauswahl getätigt, die der Server sofort annimmt, und so die vom Nutzer definierte Sprache im gesamten System bereitstellt.

Das Android System benötigt den Login nur bei der Datenübertragung. Es wird ein MD5 Hash über den Login Namen den Nutzer-ID und das Passwort gebildet und direkt als Attribut des Wurzelknotens des XML Protokolls gehangen.

Use Case Eingabe der Kundendaten: Der Nutzer gibt die Daten in die Weboberfläche ein, dabei wird er durch Auswahllisten, z.B. Land, und Vorschläge (z.B. Datum der Sammlung) vom System unterstützt.

Im Android System ist die Eingabe äquivalent zur Eingabe in die Weboberfläche,

mit dem Unterschied, dass die Daten erst in der lokalen SQLite Datenbank abgespeichert werden.

Use Case Abspeichern der Daten: Die Daten können einmal vom Webclient oder vom Androidsystem kommen. Dabei wird auch gleich die Zuordnung Kundendatensatz $\leftrightarrow$ Adresssammler gesetzt. Im Android System werden die Daten im Fehlerfall in der Datenbank gehalten, um eventuellen Übertragungsproblemen vorzubeugen.

Use Case Drucken des Gewinnloses: Nach erfolgreicher Eingabe kommt die Anzeige eines Gewinnloses. Der Losschein kann direkt über den jeweiligen zur Verfügung stehenden Drucker ausgegeben werden.

Das Android System braucht dazu den Druckaufruf über das Menü. Es sucht eine App, welche die Funktionalität für den Ausdruck bereitstellt. Dieser wird eine Formatierte Ausgabe der Daten übergeben. Die Kommunikation mit dem Drucker erfolgt dann über WLAN.

Use Case Senden der Daten: Das Android System fragt den Nutzer über die Logindaten für das Hauptsystem ab und sendet sei zusammen mit dem NutzerID verschlüsselt an das System. Um den Overhead durch Protokolldaten möglichst gering zu halten, wird es direkt mit den Daten übermittelt. So ist über die Android Systemgrenze hinaus, die Zuordnung von Adresssammler zu Kundendaten gewährleistet.

Use Case Hinzufügen neuer Veranstaltungen: Der Nutzer kann das Datum sowie die Länge der Produktvorführung einstellen. Es werden auch die Daten zum Ort der Veranstaltung eingepflegt.

Use Case Zuweisen des Verkäufers: Zum Planen der Veranstaltung gehört auch die Auswahl der geeigneten freien Verkäufer, die sie leiten. Sie werden mit der Veranstaltung verknüpft.

Zuweisen des Telefonisten: Der betreuende Telefonist wird auch mit der Veranstaltung verknüpft. Dazu wird er aus einer Liste von verfügbaren Mitarbeitern gewählt.

Use Case Sortieren der Kundendaten: Die Kundendaten liegen im Hauptsystem vor, der Nutzer kann nun mit ihm über den Webclient Interagieren und die Daten nach vorgegebenen Kriterien filtern.

Use Case Sortierte Daten Telefonisten zuordnen: Dem autorisierten Nutzer ist es möglich die gefilterten Daten oder auch Teilmengen davon Telefonisten zuzuordnen. Diese werden dann in dem zuständigen Modul weiterverarbeitet.

Use Case Anzeigen der Kundenliste: Die Telefonisten sehen nur die für sie erstellte Liste von Kunden. Sie sind nach den geführten Gesprächen geordnet. Die Kunden die einen Rückruf erwarten werden in einer separaten Liste geführt. Es werden Mitteilungen ausgegeben wenn die Rückrufzeit näher rückt.

Use Case Buchen des Termins und/oder Setzen des Status: Die Veranstaltungen können direkt über das Web Interface durch den Telefonisten gebucht werden. Dabei werden die Kunden mit den Veranstaltungen verknüpft. Es werden Zustände gesetzt, die zum Beispiel angeben, ob eine feste Zusage gegeben wurde oder ob der Interessent unentschlossen war.

Use Case Bearbeiten der Daten: Um eventuelle Änderungen oder Fehleingaben zu korrigieren, werden die Kundendaten noch einmal Editierbar angezeigt.

6.3 Datenbank

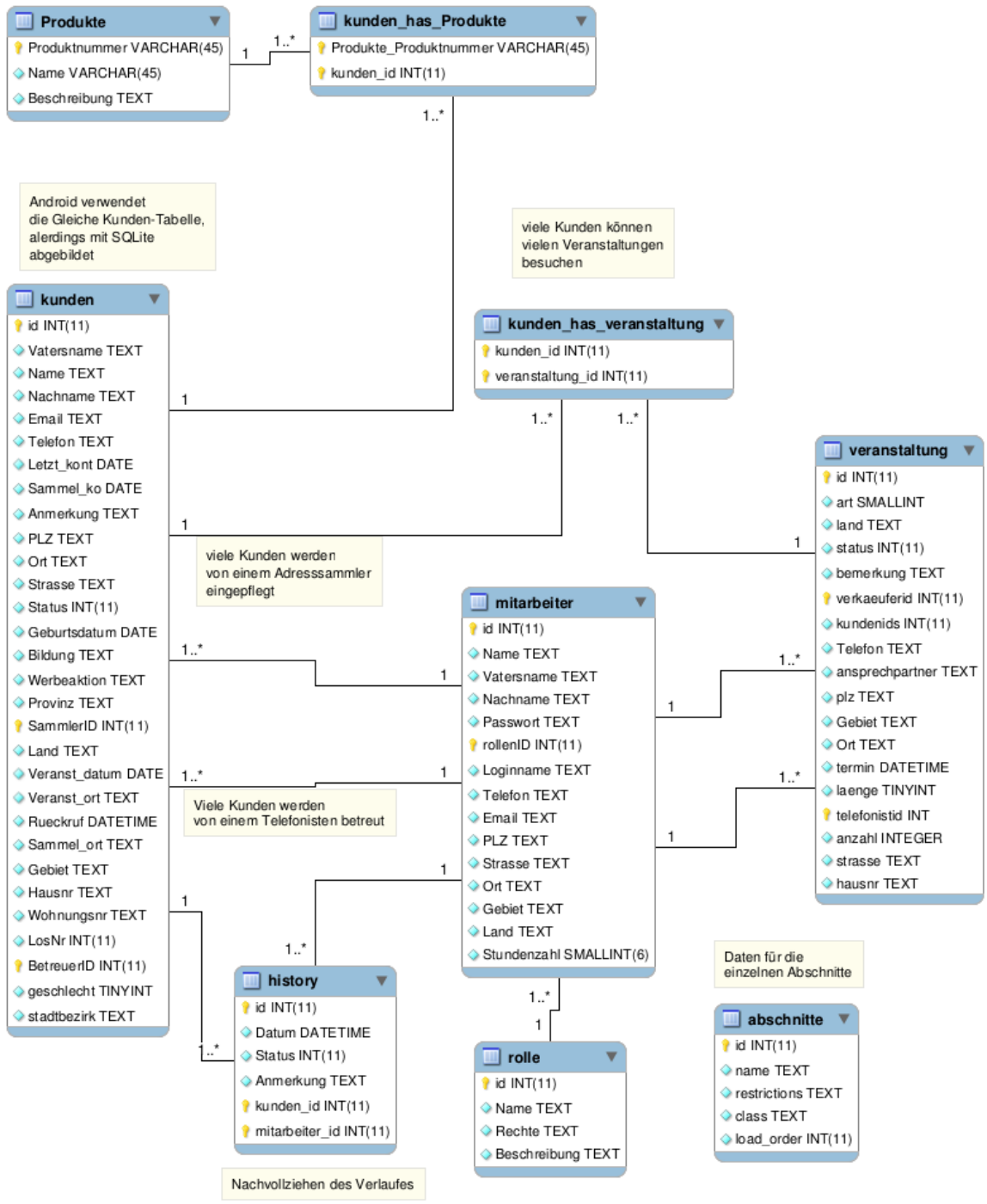

Abbildung 6.1: Datenbak Entity Relationship Diagramm

Die Datenstrukturen werden in einer MySQL Datenbank abgebildet. Hier werden Grundlagen der Normalisierung angewendet. Das Prinzip dabei ist, komplexe Beziehungen von Tabellen in einfache Beziehungen zu bringen, um Datenstrukturen zu erreichen, die stabil und flexibel gegenüber Erweiterungen des Datenmodells sind [vgl. 12, S. 116]. Die Datenstruktur der Tabelle Kunden wird so auch im Android System erstellt und ge-

pflegt. Um alle Änderungen an den Daten nachzuvollziehen wird eine Tabelle history gepflegt, die alle Verknüpfungen zu ausgeführten Aktionen aufweist und diese nachhaltig speichert.

6.4 Implementierung des Hauptsystems

Das System stützt sich auf ein Framework, welches Grundfunktionalitäten, wie dem Aufbau des Menüs und dem Bereitstellen von sicherheitsrelevanten Funktionen wie dem Login und dem Rechtemanagement. Es ist bis auf die Einsprungspunkte komplett objektorientiert und nutzt Techniken, wie Vererbung und Callbacks, um ein dynamisches Grundgerüst zu schaffen. Es gibt 3 Einsprungpunkte, welche verschiedene Daten entgegennehmen und sie an die angegebenen Abschnitte weiterreichen. Später werden sie noch ausführlicher behandelt. Jeder dieser Einsprungpunkte instantiiert zunächst die Klasse Main, welche durch Manipulation des Objektes die gewünschte Funktion bereitstellt.

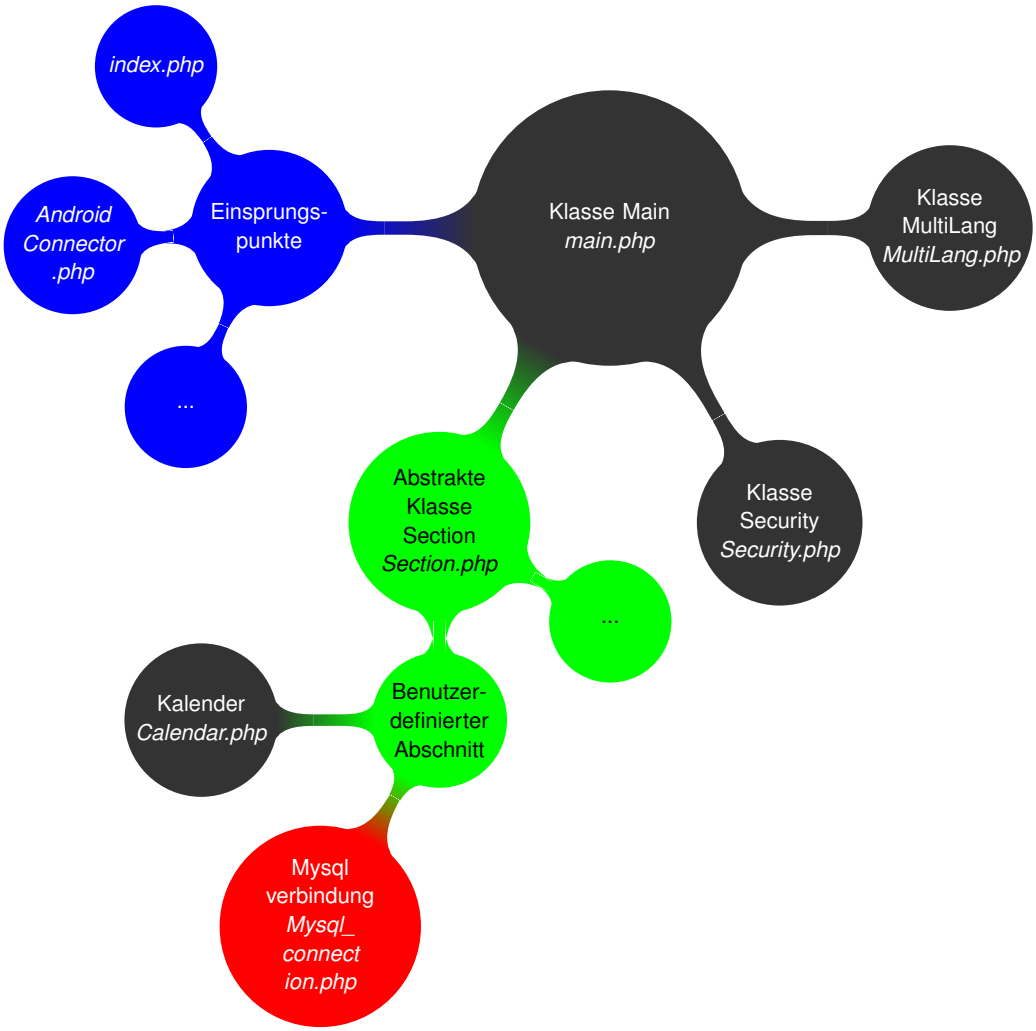

Abbildung 6.2: Aufbau des Framework

Wie in der Mindmap (Abbildung 6.2) dargestellt, besteht das Frameworks aus folgenden Komponenten:

Einsprungspunkte: Die Einsprungspunkte bieten Möglichkeiten zum Datenaustausch mit verschiedenen Techniken, Protokollen oder Endgeräten. Sie sind in der Mindmap blau dargestellt. Zu ihnen gehören die Knoten die in Tabelle 6.1 festgehalten sind. Sie sind so aufgebaut, dass sie die einzigen Interaktionspunkte mit der Webapplikation sind.

Tabelle 6.1: Übersicht der Einsprungspunkte

index.php	Allgemeiner Einsprungspunkt zur Interaktion mit der Webapplication.
AjaxRequest.php	Dieser Einsprungspunkt stellt die Ajax Funktionalitäten bereit.
AndroidConnector.php	Hier wird wie der Name schon sagt die Verbindung mit den Android Geräten aufgebaut und der Datenaustausch durchgeführt.

Hilfsklassen: Zu ihnen zählen jene Klassen, welche den einzelnen Abschnitten Funktionalitäten bereitstellen, jedoch keine Verarbeitung von Daten, die vom Nutzer oder der Datenbank bereitgestellt wurden, vornehmen. In der Mindmap gehören hierzu die Klassen Main, MultiLang, Security, sowie Kalender. Es gibt weitere dieser Klassen, jedoch spielen sie nicht so eine vordergründige Rolle für das Framework. Die Main-Klasse wird von allen Einsprunsgspunkten instantiiert. Das Objekt wird dann für die gewünschte Funktionalität beeinflusst. Es lädt auch alle Abschnitte, je nachdem, ob sie in der Rolleneinstellung des Nutzers zum Laden vorgesehen sind.

Unter Verwendung der MultiLang Klasse ist es möglich, seinen hinzugefügten Abschnitt zu übersetzen. Dies geschieht folgendermaßen, für jede Sprache liegt eine einfache Textdatei vor. Eine dieser Dateien wird je nach Sprachwahl in ein Array eingelesen. Die Array Elemente sind über definierte Konstanten abrufbar. Die Textauswahl wird durch statische Methoden getätigt.

Die Klasse Security implementiert den Login-Mechanismus der Nutzer. Hier werden auch die rechte der Rolle geladen, die der Nutzer hat. Es werden statische Funktionen bereitgestellt, die zum Beispiel die Rollenrechte des Nutzers zurückgeben.

Mit Hilfe der Kalender Klasse können Wochen oder Monatskalender realisiert werden. Die einzelnen Tage können über eine Callback Funktion mit eigenem Inhalt gefüllt werden.

Abschnitte: Sie werden in der Mindmap grün hervorgehoben. Zu ihnen gehören die Abschnittsklassen, die von der abstrakten Klasse Section abgeleitet werden und stellen die Aufbereitung, Anzeige und Erfassung der Daten bereit. Diese Klasse stellt selbst schon einige Funktionen zur Verfügung. Zum Beispiel gibt es eine Funktion zur Anzeige von Tabellen.Wobei auch hier die Zeilen mit Hilfe einer Callbackfunktion personalisiert werden können. Hier wird auch die Schnittstelle für die Abschnitte beschrieben. Sie müssen einige abstrakte Funktionen implementieren.

Datenquellen: Zu den in der Mindmap aufgezählten Klassen, welche in diese Rubrik fallen, zählt nur die Mysql_connection Klasse. Das von PHP ab Version 5.0 bereitgestellte Mysqli-Framework wurde in diese Klasse eingearbeitet. Es übernimmt den Aufbau der Datenbankverbindung. Die Mysql_connection Klasse umfasst unsichere Funktionen und sichere Funktionen zum Abschicken von Queries sowie vorgefertigte Queries zum Einpflegen, Aktualisieren oder Auslesen von Daten. Der Unterschied zwischen unsicheren und sicheren Funktionen ist, dass die sicheren Funktionen über prepared Statements realisiert werden. Der Ablauf einer solchen Abfrage ist wie folgt: In Zeile 1 wird ein neues Mysqli Objekt erstellt. Nach

Listing 6.1: Prepared Statements in PHP

```
$db = new mysqli (SQL_CONNECTION, SQL_USER, SQL_PASSW, SQL_DATABASE);
$stmt = $db->prepare ( 'QUERY' );
$stmt->bind_param('si...', $param1, $param2, ...);
$stmt->execute();
$stmt->bind_result($result1, $result2, ...);
$stmt->fetch();
```

dem die Verbindung erfolgreich hergestellt wurde, kann man eine Anfrage stellen. Zu beachten ist, das alle Eingaben durch '?' angegeben werden. Mit bind_param werden dan Referenzen an die mit '?' definierten Parameter gebunden. Der erste Parameter gibt hier die Art der Eingaben an, s-Zeichenketten, i-ganze Zahlen, usw.. Die nächste Anweisung führt die Anfrage aus. Dann müssen die Rückgabewerte wieder als Referenzen gebunden werden. Referenzen geben vereinfacht ausgedrückt nur den Speicherbereich an, in den die Werte geschrieben oder aus dem die Werte gelesen werden. Durch diese Referenzen ist es möglich, mehrere Werte in die Referenzierten Variablen zu schreiben und so das prepared Statement mehrfach zum Einpflegen von Daten zu nutzen. Am Ende werden durch fetch die in der Query abgefragten Daten in die gebundenen Rückgabereferenzen geschrieben.

6.5 Abschnitte

Im Rahmen des Projektes entstanden folgende Abschnitte, auf Grundlage des Frameworks (vgl. Abbildung 6.2):

Mitarbeiter: Dieser Abschnitt dient der Konfiguration und dem Erstellen von Rollen. Auch das Zuordnen der Mitarbeiter zu **einer** Rolle ist hier möglich. Er dient somit der Konfiguration der Rechte.

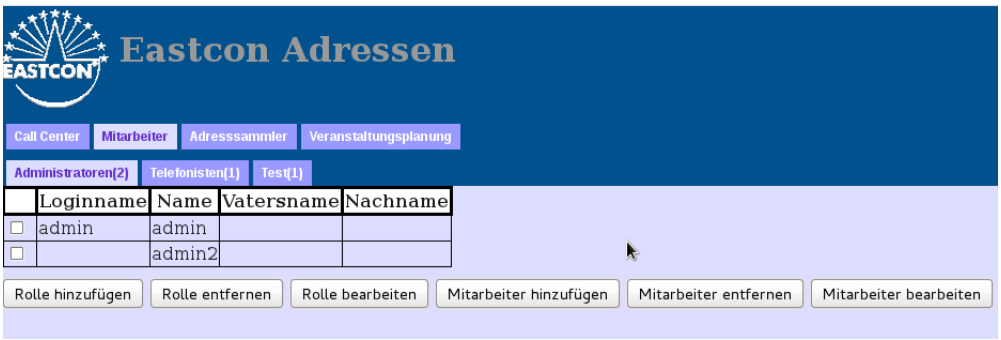

Abbildung 6.3: Abschnitt Mitarbeiter

Adresssammler: Hier können Adressen durch den Nutzer direkt in das System eingegeben werden. Da jeder Interessent eine eineindeutige Losnummer bekommen soll, auch die über das Android System eingepflegten, musste hier eine System übergreifende Lösung gefunden werden. Sie ist folgendermaßen aufgebaut:

$$\underbrace{1}_{\text{Mitarbeiter-ID}} - \underbrace{234}_{\text{Laufende ID der Kunden Tabelle}} - \underbrace{0}_{\text{Immer 0, in der Webapplication}}$$

Die Losnummer soll bei der Eingabe schon sichtbar sein, dazu ist es notwendig sie vorher zu erzeugen. Bis auf die laufende ID der Kundentabelle liegen die anderen Werte als Konstante oder seit dem Login vor. Wie man sieht kommt es hier zu einer Race Condition. Das bedeutet in dem Fall eine Resource, die nächste

Tabelle 6.2: Race Kondition der Kundeneingabe

Mitarbeiter 1 wechselt zu Abschnitt Adresssammler.	
Es wird eine Losnummer erzeugt.	Mitarbeiter 2 wechselt zu Abschnitt Adresssammler.
Eingabe der Interessentendaten.	Es wird eine Losnummer erzeugt.
Absenden und Speichern der Daten.	Eingabe der Interessentendaten.
	Absenden und Speichern der Daten.

Nummer, wird von zwei Prozessen zur gleichen Zeit genutzt. Allerdings muss hier

die Resource nicht gesperrt werden. Es sind genügend freie Nummern übrig. So muss nur eine Logik implementiert werden, um auf die nächste Verfügbare auszuweichen. Diese Zahl muss so erzeugt werden, dass die Nummer für den gerade eingebenden Nutzer geblockt wird. Dies wird folgendermaßen erreicht:

1. Eine Suche nach einen Kundendatensatz mit dem Status “Ungenutzt” und der Mitarbeiter ID des gerade aktiven Mitarbeiters.

2. Wurde ein solcher Datensatz gefunden, wird die ID dieses Datensatzes genutzt.

3. Wenn das nicht der Fall ist, wird ein neuer Datensatz mit dem Status Ungenutztünd der Mitarbeiter ID des gerade aktiven Mitarbeiters erstellt.

Vor dem Senden der Daten soll zwingend der Ausdruck des Losscheines erfolgen. Dazu wird ein Druckbutton eingefügt, der zwingend vor dem Speicherbutton betätigt werden muss.

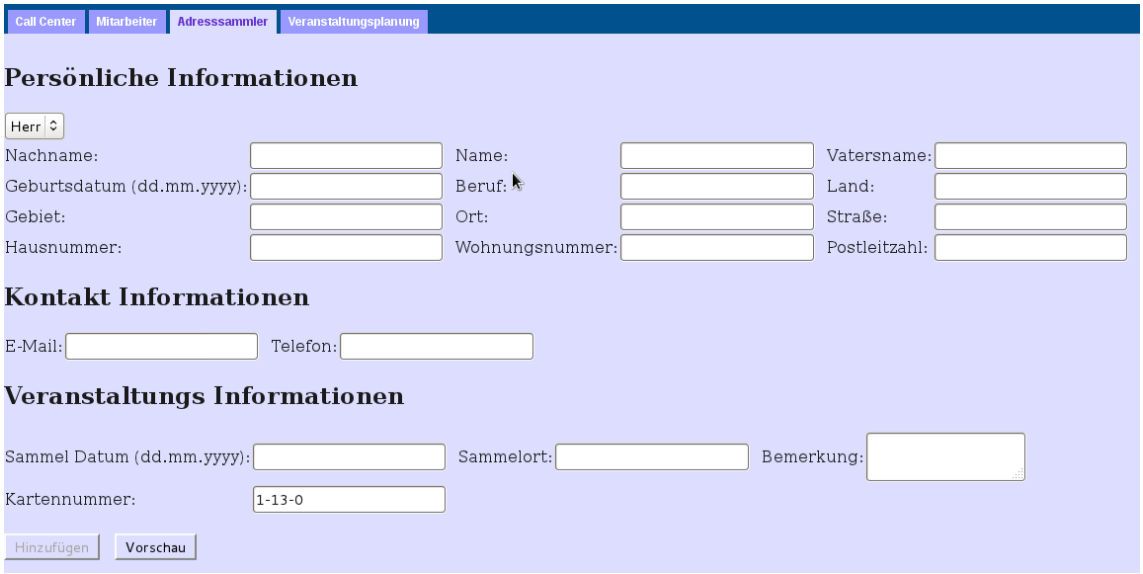

Abbildung 6.4: Abschnitt Adresssammler

Veranstaltungsplanung: Der Abschnitt beinhaltet auf der einen Seite das Zuordnen von Interessenten zu Callcentermitarbeiter, mit verschiedenen Filtern die gesetzt werden können, um die Auswahl weiter einzuengen. Die Filter funktionieren auf der Grundlage von Asynchronous JavaScript and XML (AJAX) Für die Filter von Straße und Ort wurde das JQuery Framework in Version 1.6.4 verwendet, so wie die Erweiterung Marco Polo 1.3.2. Seine Funktionsweise ist an die von Google Suggest angelehnt. Hier werden per Java Script Object Notation (JSON) die Straßen oder Orte zurückgeliefert, die mit eingegebenen Buchstaben anfangen. Nach

der Auswahl liegt dann die Liste vor. Es können auch einzelne Interessenten abgewählt werden. Die tatsächliche Anzahl wird mit angezeigt und durch AJAX oder reinem JavaScript aktuell gehalten. Auf der anderen Seite die Planung für die Veranstaltung, sie beinhaltet einen Wochenkalender in dem Veranstaltungen pro

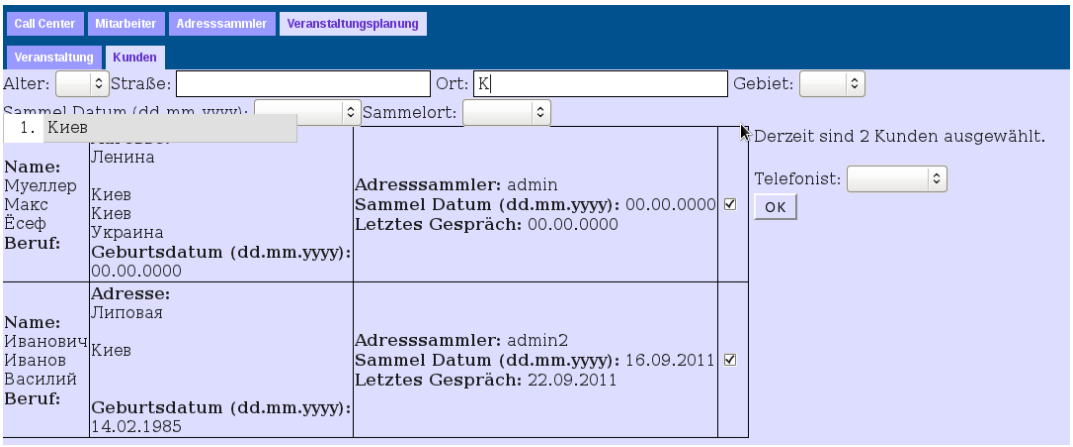

Abbildung 6.5: Abschnitt Veranstaltungsplanung Interessenten

Tag angezeigt werden. Für jeden Tag können Veranstaltungen geplant und falls gewünscht geändert werden, hier werden auch die Verknüpfungen zwischen Telefonist und Veranstaltung und Verkäufer und Veranstaltung gesetzt. Der Kalender wurde über eine Callback-Funktion für die Funktionalität des Abschnittes angepasst.

Call Center: In diesem Abschnitt werden die Interessenten am linken Rand angezeigt. Mittig ist der Kalender mit den Veranstaltungen, die der jeweilige Telefonist betreut angebracht. Die Interessenten können dabei verschiedene Status einnehmen, nach den Oben genannten Abläufen.

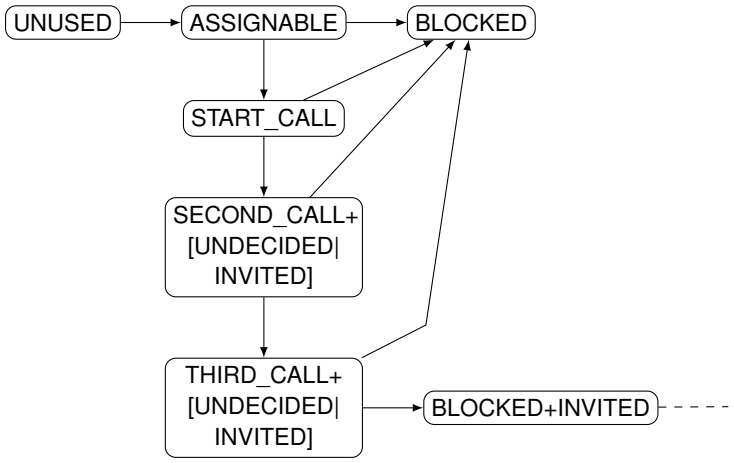

Abbildung 6.6: Zustandsdiagramm Kunden

Wie in Abbildung 6.6 dargestellt, gibt es teilweise Doppelzustände, die zusätzliche Informationen liefern. Dazu werden die Zustände binär verknüpft. Der

BLOCKED+INVITED Status gibt an, dass der Kunde zwar für wiederholte Telefonate gesperrt ist, jedoch zu einer Veranstaltung eingeladen ist. Es wird später in einem Modul für die Verkäufer fortgeführt, in diesem sollen die Verkäufer dann unter anderem Feedbacks zur Veranstaltung geben.

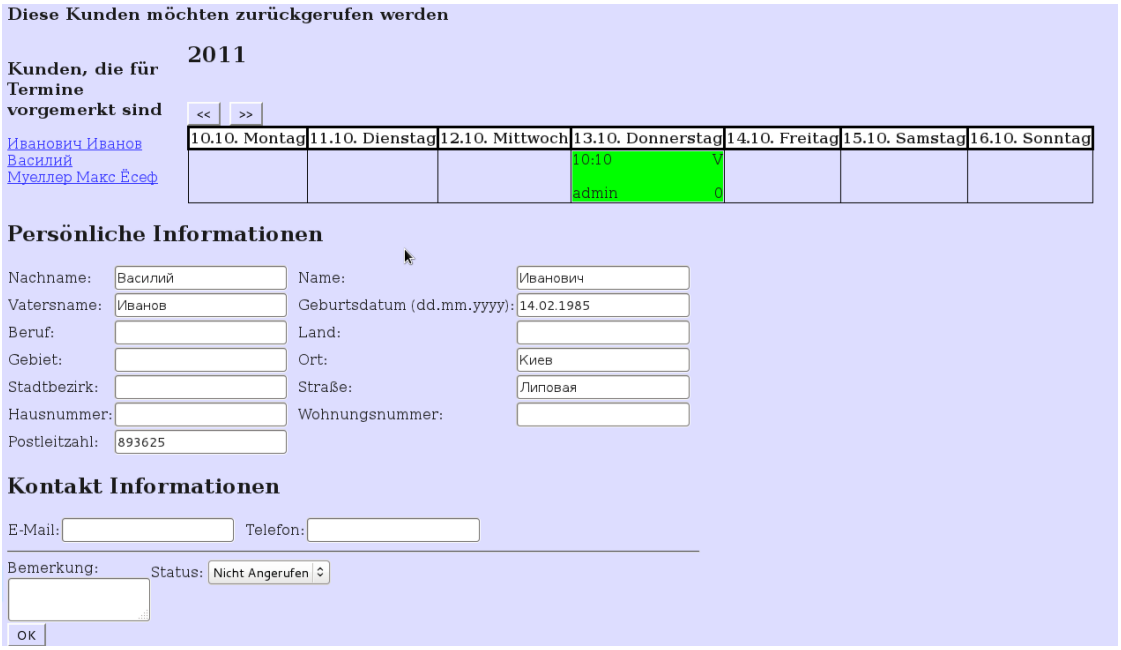

Abbildung 6.7: Abschnitt Callcenter

6.6 Implementierung des Android Systems

Das Android System beinhaltet nur eine Eingabemaske für die Daten, angelehnt an die aus Abbildung 6.4 und eine Übersicht der bisher gesammelten Daten Um die Bedienoberfläche zu erstellen gibt es zum Einen eine schwerer anpass- oder austauschbare Möglichkeit, welche zu großen Teilen dem aus Java Swing bekannten Konzepten folgt. Zum Anderen ein auf XML basierendes Konzept, womit sich die Anpassungen auf verschiedene Gerätegrößen vereinfacht. Ein beispielhafter Auszug aus der Oberfläche dieses Projektes ist in Quellcode 6.2 gegeben.

Diese Arbeit befasst sich nicht mit dem gesamten im Android möglichen Layout Einstellungen, da dies sonst den Rahmen sprengen würde. Hier wird ein relatives Layout in ein ScrollLayout gekapselt. Das ScrollLayout macht die Eingabe Scrollbar, da sie nicht auf das zum Testen vorgesehene Handydisplay passt. Das relative Layout zeigt an, dass in ihm liegende Elemente, in diesem Fall eine einfach Textanzeige (TextView) und ein Auswahlfeld (Spinner), nur relativ zu anderen angegeben werden. Das heißt, in diesem Fall, dass zuerst die Textanzeige mit an den Inhalt angepasster Höhe (*android:layout_height="wrap_content"*) und an die Fenstergröße angepasste Breite (*android:layout_width="fill_parent"*) eingefügt. Um nun das Auswahlfeld darunter einzufügen

Listing 6.2: XML Beschreibung der grafischen Nutzerschnittstelle in Android

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation ="vertical"
        android:layout_width="fill_parent"
        andr oid:layout_height ="fill_parent"
        >
        <RelativeLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        >
                <TextView
                android:id="@+id/label0"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:textSize="20px"
                android:textColor="#669"
                android:textStyle ="bold"
                android:text="Persönliche Informationen:"
                />
                <Spinner
                android:id="@+id/gender"
                android:layout_width="fill_parent"
                android:layout_height ="wrap_content"
                android:background="@android:drawable/editbox_background"
                android:layout_below="@id/label0"
                />
                ⋮
        </RelativeLayout>
</ScrollView>
```

wird einfach ein Attribut *android:layout_below* mit dem darüber (*android:id="@+id/label0"*) erzeugten ID *"@id/label0"* angegeben. Die ID's sind später auch im Programm verfügbar. Dadurch kann zum Beispiel die Wertabfrage realisiert werden. Im Java Quellcode wird das Layout auf wie in Listing 6.3 gezeigt, geladen.

Weitere Besonderheiten der Android Implementierung sind die verschiedenen Speichermechanismen, diese Arbeit geht auf zwei davon näher ein:

- SQLite Implementierung
- Shared Preferences

Erstere ist ein Datenbanktreiber, welcher eine SQLite Datenbankanbindung aufbaut. Um das zu erreichen, wird zuerst einmal die Form der Datenbank angegeben. Dies geschieht wie in Quellcode 6.4 gezeigt.

Listing 6.3: Laden der XML Beschreibung in Java

```
public class InputActivity extends Activity implements OnClickListener {

        /**Called when the activity is first created.*/
        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);

                //Setzen der Ansicht ueber die id, die aus dem
                //Namen der XML erzeugt wird
                this.setContentView(R.layout.main);

                //So wird das Spinner Objekt zur weiteren Verarbeitung geholt
                Spinner spinner = (Spinner) findViewById(R.id.gender);
                :
        }
        :
}
```

Zum Nutzen der Datenbank muss vorher feststehen, in welcher Weise sie genutzt wird. Ob zum Lesen oder Schreiben von Daten. Danach können verschiedene Funktionen dieser Objekte ausgeführt werden, um Daten zu ändern oder lesen. Es gibt einerseits Funktionen, bei denen selbst SQL Queries angeben werden können. So zum Beispiel execSQL(/*Query*/);, allerdings sind SELECT Anweisungen, also solche, bei denen man Daten abfragt hier ausgeschlossen. Auch so etwas ähnliches wie prepared Statements, nämlich precompiled Statements, sind möglich über die Funktion compileStatement(/*Statement*/); . Sie gibt ein SQLiteStatement Objekt zurück. Über dieses können Parameter gebunden werden. Es ist zu sagen, dass es im Unterschied zu prepared Statements keine Funktion gibt, um Resultate zu binden, deshalb sind für die precompiled Statements nur Aufrufe zum Einfügen, Löschen oder Aktualisieren von Daten gedacht.

Auf der anderen Seite sind auch vordefinierte Funktionen vorhanden, um zum Beispiel unter Angabe der Tabelle und der Daten eine gesamte Zeile einzufügen z.B. insert(/*Tabelle*/, /*leere Zeile*/, /*Daten*/); . Der Parameter leere Zeile gibt die Zeile an, die im Falle einer leeren Datenliste auf Null gesetzt wird. So kann eine leere Zeile eingefügt werden. Auch Funktionen, um Daten auszulesen sind vorhanden, wie zum Beispiel:

```
query(/*Tabelle*/, /*Spalte*/, /*Filter*/, /*Argumente*/, /*Gruppierung*/,
/*Gruppierungsfilter*/, /*Geordnet nach*/ );
```

Hier sind die Argumente folgendermaßen zu interpretieren:

- Tabelle: Die Tabelle aus der Daten angefordert werden.

Listing 6.4: Klasse zum nutzen einer SQLite Datenbank in Android

```
//Ableiten einer Klasse von SQLiteOpenHelper
public class AddressDB extends SQLiteOpenHelper {

        //Beschreiben der Datenbank und angabe der Version
        private static final int DATABASE_VERSION=3;
        public static final String ADDRES_TABLE_NAME = "addresses";
        private static final String ADDRES_TABLE_CREATE =
                "CREATE TABLE " + ADDRES_TABLE_NAME
                +"(id integer PRIMARY KEY AUTOINCREMENT,"
                +"Vatersname text NOT NULL, "
                :
                +"KartenID text NOT NULL)";
        private static final String DATABASE_NAME="KundenDB";

        //Sollte die Datenbank nicht vorhanden sein wird diese Funktion
        //ausgefuehrt
        @Override
        public void onCreate(SQLiteDatabase db) {
                db.execSQL(ADDRES_TABLE_CREATE);
        }

        //Sollte sich die Datenbankversion aendern, kann die
        //Datenbankstruktur hier geeandert werden
        @Override
        public void onUpgrade ( SQLiteDatabase db , int arg1 , int arg2 ) {
                throw new UnsupportedOperationException("Not supported yet.");
        }
}
```

- Spalte: Angabe der auszulesenden Spalten.
- Filter: Gibt Kriterien zur Auswahl von Zeilen an (z.B. *row='wert'*). Es wird als WHERE Klausel umgesetzt.
- Argumente: In den Filterkriterien können die Werte durch ? ersetzt werden. Dann werden die Werte dort eingesetzt.
- Gruppierung: Gibt Spalte an, nach denen die Ausgabe gruppiert wird. Es wird über eine GROUP BY Klausel abgebildet.
- Gruppierungsfilter: Werden auf die gruppierten Daten angewendet zum Beispiel: sum(row1)>row2. Diese werden als HAVING Klausel abgebildet.
- Geordnet nach: Gibt die Spalten an, nach denen in angegebener Reihenfolge geordnet wird. Abgebildet wird dies durch eine ORDER BY Klausel.

Um Konfigurationen zu speichern, ist eine einfachere Form ausreichend. Hier fiel die Entscheidung hier für die Shared Preferences Schnittstelle entschieden. Sie ist in fol-

gender Weise (vgl. Quellcode 6.5) verwendbar:

Listing 6.5: Nutzung von Shared Preferences in Android

```
/* Anfordern des SharedPrefences Objects welches fuer die
App private Daten bereitstellt */
SharedPreferences config = getPreferences(MODE_PRIVATE);
//Sofern vorhanden wird die Praeferenz UserID als Datentyp int bereitgestellt,
//ist dies nicht moeglich wird Parameter 2 zurueckgeliefert
userID = config.getInt("UserID", 0);
//Das gleiche fuer Server Praeferenz , diesmal Datentyp String
server = config.getString("Server", "http://");
```

Um nun die Daten in das Websystem zu überführen, wird ein XML basierendes Protokoll eingesetzt. Dieses Protokoll arbeitet verbindungsorientiert. Das heißt, dass zwei miteinander kommunizierende Systeme eine Verbindung aufbauen, dann Daten senden und die Verbindung wieder abbauen. Der Ablauf für das Senden der Daten des Android Systems sieht wie folgt aus: Die blaue Linie zeigt an, dass eine erneute Verbin- dung

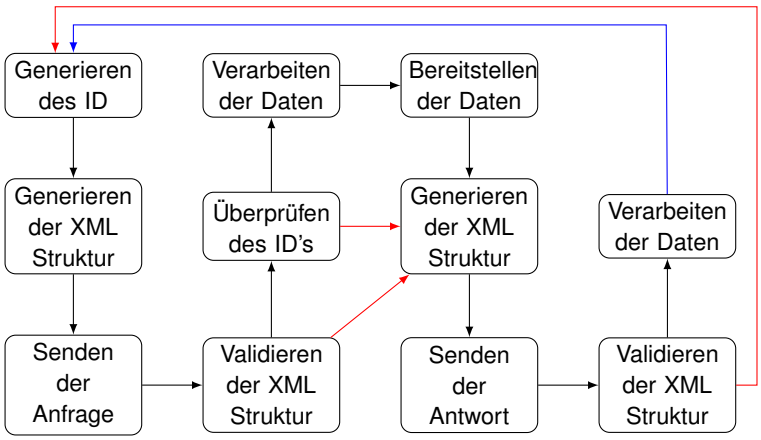

Abbildung 6.8: Abarbeitung Protokoll

aufgebaut werden kann, jedoch nicht zwingend notwendig ist. Die roten Pfade geben an, dass hier im Fehlerfall fortgesetzt wird jedoch hier mit vordefinierten Werten.

Damit das Android Programm mehrere Sprachen unterstützen kann, sind weitere Schritte nötig. Zum Einen ist es eine Anpassung der XML-Darstellung 6.2 der Benutzerschnittstelle, sie ist folgendermaßen durchzuführen. Das Attribut *android:text="text_a"*, der Textview Elemente wird abgeändert *android:text="@string/text_a_id"*. Was den mit dem *text_a_id* ID verknüpften Text lädt und an dieser Stelle einsetzt. Nun werden pro unterstützter Sprache verschiedene XML Dateien (strings.xml) erstellt, in Tabelle 6.3 Wie zu sehen ist wird an die values Verzeichnisse zur Lokalisierung ein Bindestrich einer nach ISO 369-1 vorgeschriebenen Sprachkennung. Optional kann ihr eine nach ISO ISO 3166-1-alpha-2 vorgeschriebene Regionalkennung in der Form „-r<Regionalkennung>“ folgen, um die definierten Zeichenketten im Java Quelcode nutzen zu können, wird die

Tabelle 6.3: Tabelle lokalisierte Zeichenketten in Android

Sprache	Verzeichnis	Beschreibung
Englisch(default)	*res/values/*	Die Sprache welche in einem Android Gerät welches mit Standardeinstellungen konfiguriert ist angenommen wird.
Deutsch	*res/values-de/*	Deutsche Sprachdatei, wird als Zeichenkettenquelle angenommen, wenn das Gerät auf deutsche Sprache konfiguriert ist.
Russisch	*res/values-ru/*	Lokalisierte Zeichenketten, die den russischen Sprachraum abdecken.

in der Klasse Context vorhandene Memberfunktion *getResource()* aufgerufen. Das zurückgelieferte Resource Object stellt nun die Funktion *getString()* mit dem Integerparameter id bereit, die den mit Angabe des ID's die gewünschten lokalisierte Zeichenkette zurück liefert.

Die im Abschnitt 6.4 vorgestellte Racecondition wird im Android System folgendermaßen aufgelöst:

$$\underbrace{1}_{\text{Mitarbeiter-ID}} - \underbrace{234}_{\text{Laufende ID der Kunden Tabelle}} - \underbrace{0}_{\text{wird Berechnet}}$$

Der berechnete Wert am Ende besteht aus der Summe der letzten drei zweistelligen Hexadezimalzahlen der MAC Adresse der WLAN-Schnittstelle des jeweiligen Gerätes.

7 Fazit

Es entstand eine Software welche den Ablauf vom Sammeln der Adressen bis hin zur Befüllung der Veranstaltung abbildet. Durch zeitliche Engpässe und Anforderungen die sich erst im Verlauf der Entwicklung zeigten konnten, nicht alle Funktionen eingebaut werden. Auch ein umfassender Test der bisher implementierten Module war noch nicht möglich.

Die Zeit die für das Projekt und das Anfertigen dieser Arbeit war knapp bemessen. Dies ist zu großen Teilen dem Umfang der Implementierung zu schulden. Es wurden teilweise viele zeitliche Ressourcen verbraucht um Kernfunktionen der Module zu erstellen. Einige dieser Funktionen sind zur Zeit noch unzureichend in der Software implementiert, werden aber zu einem späteren Zeitpunkt noch einmal überarbeitet. Im Rückblick war das Projekt zu umfangreich.

Das betriebliche Umfeld beeinflusste mich positiv beim Erstellen dieser Arbeit. Es wurden auch ständig neue Anregungen gegeben die Software zu verbessern oder an bestehende Abläufe anzupassen.

Ausblick

Die entstandene Software weißt noch Mängel auf, sie ist nicht bereit für den produktiven Einsatz. In weiteren Iterationen sind diese Mängel zu beseitigen. Weitere Module (z.B. für Verkäufer) sind in Planung. Das entstandene Framework sollte hinsichtlich Oberflächengestaltung überarbeitet werden, um größere Akzeptanz von Nutzern zu bekommen. Dabei sollten die Module sich diesen Designvorschriften anpassen. Das entstandene System wird Fertiggestellt und soll später in den produktiven Einsatz kommen. Eine Anbindung an SugarCRM ist auch geplant.

Literaturverzeichnis

[1] ARM LTD. (Hrsg.): *ARM Prozessor Architektur Jazelle abschnitt*. http://www.arm.com/products/processors/technologies/jazelle.php, Abruf: 28. Aug. 2011

[2] CASTELEYN, Sven ; DANIEL, Florian ; DOLOG, Peter ; MATERA, Maristella: *Engineering Web Applications*. Springer-Verlag Berlin Heidelberg 2009, 2009. – ISBN 987

[3] CHRISTIAN HÜGEL, SEBASTIAN VAHL (Hrsg.): *Fedorawiki Iptables*. http://www.fedorawiki.de/index.php/Iptables, Abruf: 28. Aug. 2011

[4] CONTRIBUTORS TO THE UBUNTU DOCUMENTATION WIKI (Hrsg.): *Tasksel-Comunity Ubuntu Documentation*. https://help.ubuntu.com/community/Tasksel, Abruf: 28. Aug. 2011

[5] DAN WALSH (Hrsg.): *Manpage SELinux*. http://linux.die.net/man/8/selinux, Abruf: 28. Aug. 2011

[6] GOOGLE INC. (Hrsg.): *Google Android Dev Guide*. http://developer.android.com/guide/index.html, Abruf: 28. Aug. 2011

[7] HIPPE, Hajo ; WILDE, Klaus D.: *Grundlagen des CRM: Konzepte und Gestaltung*. 2. Auflage. Betriebswirtschaftlicher Verlag Dr. Th. Gabler/GWV Fachverlag GmbH, 2006. – ISBN 3–409–22518–8

[8] HOCKMANN, Volker ; KNÖLL, Heinz-Dieter: *Profikurs Sicherheit von Web-Servern*. 1. Auflage. Vieweg+Teubner Verlag | GWV Fachverlag GmbH Wiesbaden, 2008. – ISBN 978–3–8348–0022–0

[9] JAMES MORRIS (Hrsg.): *SELinux Wiki*. http://selinuxproject.org/page/Main_Page, Abruf: 28. Aug. 2011

[10] JAMES MORRIS (Hrsg.): *SELinux Wiki*. http://selinuxproject.org/page/Main_Page, Abruf: 28. Aug. 2011

[11] JEAN-PHILIPPE LANG (Hrsg.): *Suricata Wiki*. https://redmine.openinfosecfoundation.org/projects/suricata/wiki, Abruf: 28. Aug. 2011

[12] KANNENGIESER, Caroline ; KANNENGIESER, Matthias: *PHP^5/MySQL^5*. Master Edition. Franzis Verlag GmbH, 2007. – ISBN 987–3–7723–7110–3

[13] LINUX NEW MEDIA AG (Hrsg.): *Etckeeper stellt Systemkonfiguration unter Versionskontrolle « Online Artikel « Linux-Magazin Online*. http://www.linux-magazin.de/Online-Artikel/Etckeeper-stellt-Systemkonfiguration-unter-Versionskontrolle, Abruf: 28. Aug. 2011

[14] NATIONAL SECURITY AGENCY (Hrsg.): *Security-Enhanced Linux - NSA/CSS*. http://www.nsa.gov/research/selinux/, Abruf: 28. Aug. 2011

[15] NETCRAFT LTD (Hrsg.): *June 2010 Web Server Survey | Netcraft*. http://news.netcraft.com/archives/2010/06/16/june-2010-web-server-survey.html, Abruf: 28. Aug. 2011

[16] NETFILTER WEBMASTER (Hrsg.): *netfilter.org ïptablesppproject*. http://www.netfilter.org/projects/iptables/index.html, Abruf: 28. Aug. 2011

[17] ORACLE CORPORATION AND/OR ITS AFFILIATES (Hrsg.): *MySQL Allgemeine Sicherheitsrichtlinien*. http://dev.mysql.com/doc/refman/5.1/de/security-guidelines.html, Abruf: 28. Aug. 2011

[18] PHP DOCUMENTATION GROUP (Hrsg.): *PHP: Einführung - Manual*. http://www.php.net/manual/de/oop5.intro.php, Abruf: 28. Aug. 2011

[19] PHP DOCUMENTATION GROUP (Hrsg.): *PHP Prepared Statements und Stored Procedures*. http://www.php.net/manual/de/pdo.prepared-statements.php, Abruf: 28. Aug. 2011

[20] RED HAT, INC. AND OTHERS. (Hrsg.): *Administration Guide Draft/Apache - Fedora- Project*. http://fedoraproject.org/wiki/Administration_Guide_Draft/Apache, Abruf: 28. Aug. 2011

[21] RED HAT, INC. AND OTHERS. (Hrsg.): *Features/systemd - FedoraProject.* http://fedoraproject.org/wiki/Features/systemd, Abruf: 28. Aug. 2011

[22] RED HAT, INC. AND OTHERS. (Hrsg.): *Foundations - FedoraProject.* https://fedoraproject.org/wiki/Foundations, Abruf: 28. Aug. 2011

[23] RESCORLA, E.: HTTP Over TLS / Internet Engineering Task Force. Version: Mai 2000. http://www.rfc-editor.org/rfc/rfc2818.txt. 2000 (2818). – RFC. – 7 S.

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Mittweida, 12. Dezember 2011